UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIONES, SONIDO E IMAGEN

Departamento de Teoría de la Señal y Comunicaciones

PROYECTO FIN DE CARRERA

DESARROLLO DE UN MÉTODO DE FUSIÓN DE REGIONES PARA SEGMENTACIÓN DE IMÁGENES

Autor: Lara Fajardo Ibáñez

Tutor: Jesús Cid Sueiro

Madrid, Abril de 2009

¿Me podrías indicar, por favor, hacia dónde tengo que ir desde aquí?
- Eso depende de a dónde quieres llegar -contestó el Gato.
- A mí no me importa demasiado a dónde... -empezó a decir Alicia.
- En ese caso, da igual hacia dónde vayas - interrumpió el Gato.
- ...siempre que llegue a alguna parte -terminó Alicia a modo de explicación.
- ¡Oh! Siempre llegarás a alguna parte -dijo el Gato- si caminas lo bastante.

Alicia en el país de las maravillas, de Lewis Carrol

Agradecimientos

Este trabajo va a dedicado a todas aquellas personas que me han dado su apoyo y han creído en mi.

En especial a mis padres, por su inmensa paciencia y compresión, porque sin ellos no soy nadie y no hubiera llegado a ninguna parte.

A mis cuatro hermanos, que me han hecho el camino fácil y tantas cosas me han enseñado. A Ángela y Ricardo.

A esos pequeños, Jesús y Ángela, que añoro no poder verlos crecer.

A Manu, que en estos últimos años ha sido mi apoyo diario y me ha ayudado a ser fuerte.

A mis amigas de la infancia, Tania, Sofía y Ascen, que me han hecho reír en tantas ocasiones y han llenado mi vida de historias.

A todos mis compañeros que he conocido durante la carrera. A Adri, Elena, Debora, Raúl y Amanda, que me han enseñado mucho y me han dado su amistad. De todos ellos, quiero mencionar especialmente a Elena, muchas gracias por el tiempo que me has dedicado y porque tú también formas parte de este estudio.

A mi tutor, Jesús, muchas gracias por ayudarme y guiarme durante este trabajo.

A ellos y a todos aquellos que me acompañan, o me han acompañado, gracias.

Resumen

Este proyecto consiste en el desarrollo y la implementación de un bloque de post-procesamiento de regiones en imágenes segmentadas. El principal objetivo es reducir la sobre-segmentación de regiones, características de algunos algoritmos de de segmentación, mediante mecanismos de fusión de regiones.

El sistema de post-procesamiento de imágenes permite la extracción de características internas de las regiones. Los parámetros disponibles son el número de componentes conectadas, la compacidad, la convexidad, la rectangularidad y excentricidad. A partir de estos descriptores, se estudia si es conveniente o no realizar una fusión y, si procede, realizar la fusión de regiones.

El resultado final del post-procesado es una imagen dividida en regiones unificadas y no dispersas.

Palabras clave: post-procesamiento de imágenes, fusión de regiones, extracción de características, descriptores de región, sobre-segmentación.

Abstract

This project consists of the development and the implementation of a system of image processing. The main objective is reducing the region over-segmentation, improving the existing segmentation algorithms.

The system of image post-processing allows the extraction of internal characteristics of the segmented image regions. Parameters available are the number of connected components, compactness, convexity, bounding box parameters and eccentricity. From these descriptors, the convenience or not to make a fusion is analyzed and, if so, the region-based fusion is made.

The final result of the post-process is an image divided in unified and non-dispersed regions.

Keywords: image post-processing, region-based fusion, region descriptors, over-segmentation.

Índice de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1

INTRODUCCIÓN

Este capítulo se dedica a la presentación resumida de este proyecto. Describiremos la motivación y los objetivos que se presentaron al comienzo de este trabajo. Finalmente, se describirá la estructura del documento.

1.1 MOTIVACIÓN

En el tratamiento digital de las imágenes se suelen distinguir tres procesos, que en ocasiones se solapan, que son *Procesamiento*, *Análisis* y *Aplicaciones*. El procesamiento implica la manipulación de las imágenes vistas como señales digitales, para extraer la información más elemental subyacente. El análisis se encamina a determinar ciertas estructuras elementales tales como bordes o regiones así como las relaciones entre ellas y, finalmente, las aplicaciones tratan de dar solución a los problemas relacionados con ciertas situaciones del mundo real como: reconocimiento, movimiento, reconstrucción 3-D, etc. (*Ref*. [1]). El proceso general se puede sintetizar en la figura 1.1, en la que las cajas representan datos y las burbujas procesos.

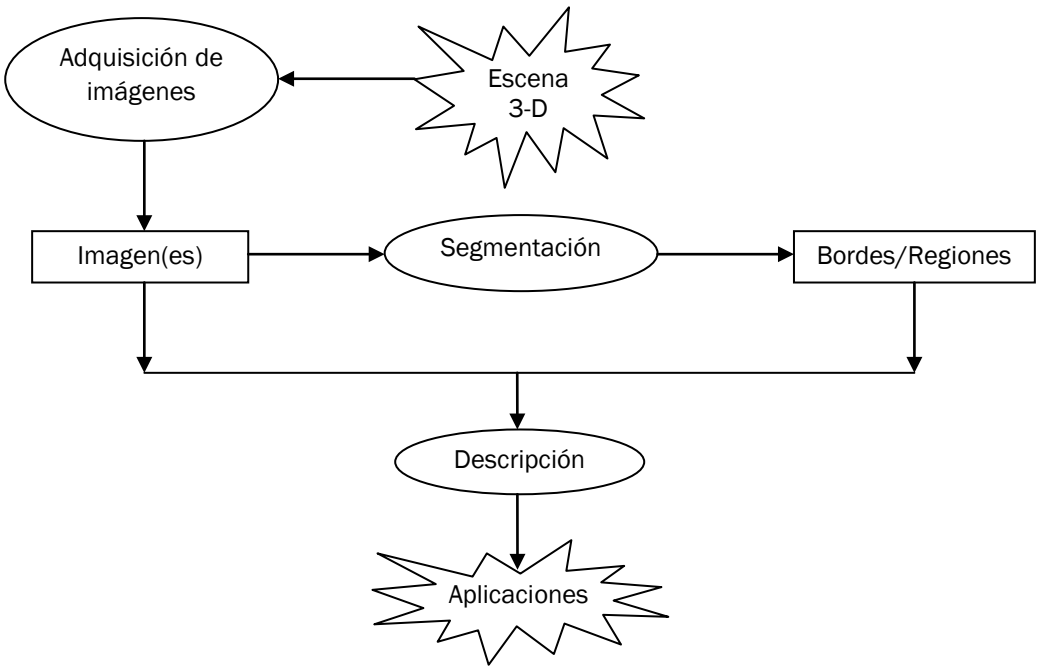


Figura 1.1. Diagrama de bloques mostrando el proceso completo del tratamiento digital de imágenes (*Ref. [1]*).

Como podemos observar en la figura anterior, la segmentación es una parte del análisis en el tratamiento digital de imágenes. Las técnicas de segmentación dividen una imagen en regiones con características similares, cada una de estas regiones se denominan objetos.

La segmentación de imágenes es uno de los aspectos más importantes de la percepción visual humana. Las personas usamos nuestro sentido visual para dividir el entorno en diferentes objetos y, así, reconocerlos. Desafortunadamente, no es fácil crear algoritmos artificiales cuya actuación sea comparable a la del sistema visual humano.

La segmentación suele ser la primera tarea de cualquier sistema de procesado de imágenes. Pero, en muchas ocasiones estas técnicas no son suficientes, causan sobre-segmentación y se necesitan otras técnicas que mejoren estos resultados. Este proceso recibe el nombre post-procesado de imágenes.

El post-procesado trata de unificar en una sola región aquellas regiones que han sufrido sobre-segmentación, para que así la percepción sea lo más parecida a la del sistema visual humano.

El post-procesado realiza una descripción de las regiones extraídas en la segmentación. Esta descripción se puede diferenciar en *descriptores de contorno*, cuyo objetivo es la identificación de los bordes mediante ajuste de rectas, curvas funciones polinómicas, códigos encadenados, etc. y *descriptores de región*, encaminados a obtener propiedades tales como color, textura, superficie, nivel medio de intensidad, etc.

Así surge este proyecto, con el propósito de mejorar las técnicas de segmentación realizadas en otros proyectos fin de carrera por alumnos en el departamento de Teoría de la Señal y Comunicaciones de la Universidad Carlos III de Madrid.

En conclusión, este proyecto tiene por fin la continuación del desarrollo de un proyecto de algoritmos de segmentación iniciado en [25]. Asimismo, este proyecto servirá como desarrollo para futuros trabajos como puede ser la mejora de los algoritmos realizados o con la continuación dentro del proceso de tratamiento digital de imágenes.

1.2 OBJETIVOS

El principal objetivo de este proyecto es la creación e implementación de una herramienta software que realice el trabajo de un bloque de post-procesado de imágenes basado en la descripción de regiones. Se trata de desarrollar un bloque flexible y modular que sea factible con cualquier algoritmo de segmentación. Es decir, que no dependa de las características que se obtienen de la imagen en la segmentación sino que realice su tarea sin tener que ejecutar un algoritmo de segmentación en concreto.

En primer lugar, se realizará una combinación de posibles fusiones de regiones de las que se realizará, posteriormente, una extracción de características.

Después, se estudiarán las regiones que se han obtenido de la segmentación y de las posibles combinaciones calculadas por el primer algoritmo. Este estudio consistirá en extraer características a partir de una descripción de contorno y de regiones de la imagen, con parámetros tales como el número de componentes conectadas, la convexidad, compacidad, rectangularidad y excentricidad.

En función de estos parámetros y, en base a un umbral, se decidirá si es conveniente o no realizar la fusión de determinadas regiones. Y si se considera beneficioso se procederá a la fusión de las regiones, obteniendo así una nueva imagen.

El fin de estos algoritmos se dará cuando ya no se considere necesario la obtención de más fusiones y el resultado final será una imagen divida en regiones mucho más compactas que la imagen obtenida de la segmentación.

1.3 ESTRUCTURA DEL DOCUMENTO

Esta memoria se divide en seis capítulos que explican el trabajo realizado en este proyecto.

En el *Capítulo 1* se cuenta la introducción al post-procesamiento de imágenes así como los objetivos que se marcaron para la realización de este proyecto.

A continuación, en el *Capítulo 2*, trataremos de introducir al lector en el ámbito del post-procesado de imágenes. Este capítulo recibe el nombre de *Estado del Arte* y en él se cuentan los descriptores más usados para la extracción de características, así como los últimos desarrollos alcanzados en algoritmos para el post-procesamiento de imágenes.

Más adelante, hablaremos del desarrollo del proyecto que está dividido en dos capítulos. En el *Capítulo 3*, explicaremos el diseño algorítmico del trabajo realizado en la herramienta software mientras que, en el *Capítulo 4,* trataremos de explicar la realización de la herramienta software.

En el *Capítulo 5*, evaluaremos los algoritmos diseñados para la aplicación con una serie de pruebas. En el primer apartado de este capítulo realizaremos unos experimentos subjetivos con la calidad de los parámetros y la mejora que supone el bloque de post-procesado en la segmentación. En el siguiente apartado, evaluaremos los algoritmos de post-procesado de forma automática.

Finalmente, concluirá esta memoria con el *Capítulo 6* en el que expondremos una serie de conclusiones extraídas de la realización, implementación y evaluación del proyecto junto a unas líneas futuras para la mejora del trabajo realizado.

Como ayuda a la comprensión de la herramienta, se incluyen dos anexos. En el *Anexo A* se hace una revisión a la estructura del código para que aquel que tome la herramienta como punto de partida para un nuevo proyecto sea capaz de entender el código programado. En el *Anexo B* se incluye un manual de usuario de

la interfaz gráfica para aquel que quiera hacer uso de la herramienta sin tener conocimientos de programación.

Al final del documento, se incluye la *Bibliografía* utilizada durante la realización del desarrollo de la herramienta software así como la utilizada durante la elaboración de esta memoria.

CAPÍTULO 2

ESTADO DEL ARTE

En este capítulo vamos revisar el problema del post-procesado de regiones en segmentación. Hablaremos de los parámetros usados en el desarrollo de los algoritmos de la herramienta software así como de los algoritmos ya desarrollados por otros autores para el post-procesamiento de imágenes. Ambas partes nos servirán, posteriormente, para entender la herramienta desarrollada en este proyecto.

2.1 DESCRIPCIÓN DE REGIONES

2.1.1 Introducción

Después de que una imagen es segmentada en regiones, se necesita realizar una representación y descripción más específica de estas regiones. Una representación matemática que permita describir el objeto de forma más compacta. Esto consiste en extraer las características de la región mediante un vector, denominado vector de características, que cuantifica el valor de las mismas.

Una región se puede representar mediante sus características externas o por sus características internas, habitualmente se les denomina descriptores de contorno y descriptores de región, respectivamente. La elección de una técnica u otra se basa en los resultados que queremos obtener. En un gran número de ocasiones es conveniente que los descriptores, sean simples, precisos y con gran poder de discriminación, sean invariantes, si es posible, ante escalado, posición y rotación del objeto en la imagen.

Los descriptores de contorno tratan de describir la región aproximándola mediante una sucesión de líneas y puntos, asemejándola a una figura. Mientras que, en el estudio de las características internas estudiamos los píxeles que comprenden una región.

En el presente capítulo, haremos una explicación de los descriptores de contorno y de los descriptores de región, exponiendo los parámetros utilizados en el desarrollo de la herramienta software.

Las referencias usadas para este apartado del capítulo, principalmente, son [1], [2], [3], [9] y [10].

2.1.2 Descriptores de contorno

La representación del contorno de un objeto segmentado en la imagen es el conjunto de píxeles que definen dicho contorno.

Muchos de los descriptores de este tipo se basan en:

- **Los códigos de cadena**. Son usados para representar la frontera en a base a un conjunto de segmentos, de una longitud fija y dirección específica, conectados entre sí.
- **Representación polar o signatura**. La idea de esta técnica se basa en representar el contorno como una función polar unidimensional.
- **Descriptores de Fourier**. La transformada discreta de Fourier (DFT) permite extraer las componentes en frecuencia de una curva discreta periódica cualquiera, como puede ser un contorno. Los descriptores de Fourier son el conjunto de puntos en el espacio de frecuencias, resultantes de transformar mediante la DFT dichos puntos del contorno.

En el software desarrollado no hemos usado ninguno de estos descriptores ya que esta representación no es cómoda por varios motivos. Fundamentalmente, son descriptores muy sensibles al ruido y a los defectos procedentes de la segmentación, además no son nada eficientes desde el punto de vista computacional y de almacenamiento de la información.

2.1.2.1 Algunos descriptores básicos

Además de todos estos descriptores complejos, podemos encontrar otros más simples, como son los parámetros geométricos. Dentro de los descriptores de contorno, encontramos el perímetro o la curvatura.

El **perímetro** viene dado por la longitud de su código de cadena (con ocho direcciones) pero ponderando los pasos diagonales por $\sqrt{2}$ y los horizontales y verticales por 1, es decir, viene dado por el número total de píxel que configuran su contorno pero los píxeles de bordes diagonales se ponderan con $\sqrt{2}$. [9]

El **diámetro** de un contorno viene dado por la distancia Euclídea entre los dos píxeles del contorno más alejados. Se define como,

$$Diam(B) = \max_{i,j}[D(p_i, p_j)] \qquad (2.1)$$

donde *D* es la distancia y p_i y p_j son los puntos del contorno. Dichos puntos no son siempre únicos, como ocurre en una circunferencia, pero es un descriptor de interés cuando sí lo son. La recta que pasan por dichos puntos se llama **eje mayor** de la región. El rectángulo, con dos lados paralelos al eje mayor, que tiene la propiedad de que es el menor rectángulo que contiene al contorno se llama **rectángulo base** o "**bounding box**" (Figura 2.1). El cociente entre la longitud del lado mayor y la longitud del lado menor se llama **excentricidad** del contorno.

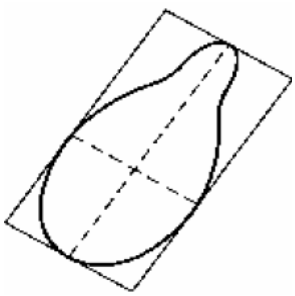


Figura 2.1. Rectángulo base de un contorno (*Ref. [2]*).

El **centro de gravedad** $(\bar{x}, \bar{y})$, o **centroide**, de un contorno es un único punto representativo de la región determinado por un conjunto de píxeles $\{(x_i, y_i), i = 1, 2, \ldots, N\}$ y se calcula mediante,

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \quad \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N} \qquad (2.2)$$

El **eje menor** del contorno viene definido por la recta perpendicular al eje mayor que pasa por el centro de gravedad del contorno.

Todos los parámetros simples, definidos anteriormente, se caracterizan por ser invariantes a traslaciones pero, no frente a escalado.

Otro parámetro básico es la **curvatura**. Se define como la tasa de cambio de la pendiente de la frontera. En general, es bastante difícil conseguir una curvatura fiable ya que suelen ser localmente desiguales. Por ello, se calcula mediante aproximación poligonal o a partir de versiones suavizadas. Consiste en utilizar la diferencia de las pendientes de segmentos adyacentes del contorno.

Los descriptores usados, de esta categoría, en la herramienta desarrollada son la excentricidad y la rectangularidad. Aunque como se ha explicado anteriormente, estos descriptores indirectamente utilizan otros como los ejes y el perímetro. Esta elección se basa en la sencillez así como en el bajo coste computacional.

2.1.3 Descriptores de región

Como ya se ha mencionado anteriormente, estos descriptores hacen un estudio de los píxeles que conforman una región.

Dependiendo del criterio a utilizar, una región puede describirse de diversas maneras. Puede ser vista, como un conjunto de puntos conectados entre sí, es decir, partiendo de un punto de la región podemos llegar a otro punto de la misma sin abandonar dicha región, o puede ser descrita por el número de huecos que presenta, son en ambos casos *propiedades topológicas*. Por el contrario, si podemos obtener valores tales como, por ejemplo, el área, estaremos refiriéndonos a *propiedades métricas*. Pero, también es posible fijarnos en *irregularidades* de las regiones para su diferenciación. A continuación, pasamos a estudiar estos enfoques.

2.1.3.1 Propiedades topológicas

Las propiedades topológicas son normalmente usadas para realizar una descripción global de la región en el plano de la imagen. Se caracterizan por ser invariantes a deformaciones del plano de la imagen. Los descriptores topológicos más importantes son el número de componentes conectadas y el número de Euler.

Una **componente conectada o conexa** de un conjunto es un subconjunto de dimensión máxima tal que cualesquiera dos de sus puntos pueden unirse por una curva continua (sin rupturas) perteneciente enteramente al subconjunto.

Si *p* y *q* son píxeles de un subconjunto *S* especificado de la imagen. Se dice que *p* está conectado a *q* dentro de *S* si existe un camino entre ellos que consista totalmente de píxeles de *S*. Para cualquier píxel *p* dentro de *S*, el conjunto de píxeles de *S* conectados a *p* se denomina componente conectada de *S*. Por tanto, cualquier par de píxeles de una misma componente conectada están conectados entre sí, y componentes conectadas son disjuntas.

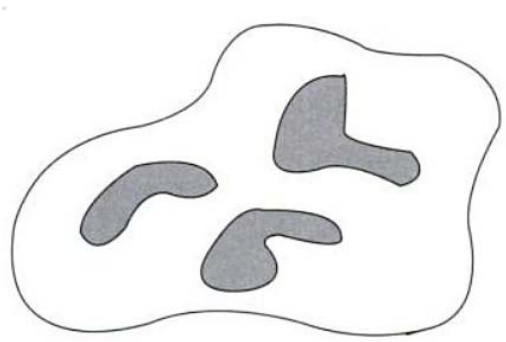


Figura 2.2. Una figura con tres componentes conectadas (*Ref. [2]*).

La capacidad de asignar etiquetas diferentes a las distintas componentes conexas disjuntas de una imagen tiene una importancia fundamental en el análisis automatizado de la imagen.

Esta técnica se ha utilizado como descriptor dentro del vector de características desarrollado en la herramienta, debido a que nos da una idea general de cómo se sitúan las diferentes regiones de la imagen y si es conveniente su fusión.

El etiquetado de componentes conectadas asigna una etiqueta única a un subconjunto de puntos llamados comúnmente objetos. La condición para pertenecer a cierta región es que el pixel en cuestión esté conectado con la región. La conectividad se refiere al hecho de que dos pixeles se encuentren unidos [8]. Así pues, se fusionarán aquellas regiones que, en su conjunto, tengan un mayor

número de componentes conectadas ya que de esta manera, se consigue una imagen más compacta y, por tanto, una región más compacta.

Otra medida topológica bastante utilizada para un conjunto de regiones conectadas, algunas de las cuales pueden tener huecos, es el **número de Euler** (*E*) como descriptor. El número de huecos (*H*) y el número de componentes conectadas (*C*) se usa para definir esta propiedad. Así pues, el número de Euler se calcula:

$$E = C - H \qquad (2.3)$$

Este número es invariante frente a traslaciones, rotaciones y cambios de escala, y nos permite de forma sencilla discriminar entre ciertas clases de objetos.

A continuación, vemos un ejemplo. En la figura 2.3, la letra *A* tiene un número de Euler igual a cero, ya que el número de componentes conectas es uno y el número de huecos es 1. Mientras que para la letra *B* su valor es de menos uno (C=1, H=2).

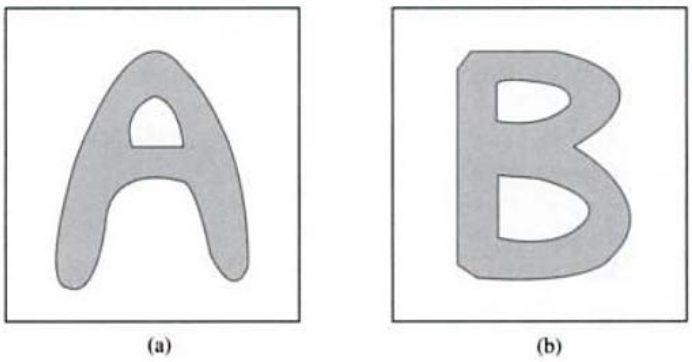


Figura 2.3. Regiones con el número de Euler igual a (a) 0 y (b) -1. (*Ref. [2]*).

2.1.3.2 Propiedades métricas

Las propiedades métricas de las figuras dependen de una métrica. Conceptualmente, las métricas son generalizaciones de la distancia Euclídea, así una propiedad métrica cambiará si el plano de la figura se distorsiona.

A partir de los descriptores métricos más simples, como son el área, el perímetro o el centro de gravedad, se pueden calcular parámetros más complejos que nos ayudaran en la unión de regiones.

El **área** de una región se define como el número de píxeles contenidos dentro de su frontera. Se puede obtener también de forma sencilla a partir de su código de cadena, mediante el uso del perímetro.

Otro parámetro es el **centro de gravedad o centroide** $(\bar{x}, \bar{y})$, es un punto único de la región, que se calcula como se ha explicado anteriormente,

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \quad \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N} \qquad (2.4)$$

Obsérvese que el centroide de una región no tiene por qué coincidir con el centroide de su contorno.

Aunque a veces el área y el perímetro se utilizan como descriptores, se suelen utilizar en los casos en los que el tamaño de la región de interés sea invariante.

Existen otros parámetros, que se usan con mayor frecuencia, como la **compacidad o circularidad de una región**, que se define como,

$$c = \frac{A}{P^2} \qquad (2.5)$$

donde *A* es el área y *P* es el perímetro de la región. El perímetro se eleva al cuadrado para obtener un parámetro adimensional.

Su valor máximo corresponde a los círculos (máxima superficie para un perímetro dado) y vale 1/(4π) (≈0.07957), por ello es una medida de circularidad; para el triángulo equilátero vale 1/(123) (≈0.048) y para el cuadrado vale 1/16 (=0.0625).

Los valores pequeños del parámetro indican objetos alargados. Además, es invariante frente a traslaciones, giros y escalado. Conviene normalizar sus valores

al intervalo [0,1] dividiendo por 1/(4π) para garantizar que el valor que le asigne al círculo sea 1, y un valor menor a cualesquiera otra figura geométrica, tanto menor cuanto más alargada sea o presente espículas [9]. Para conseguir esto, se define otro parámetro llamado **tasa circular o "circularity ratio"**,

$$R_c = 4\pi \frac{A}{P^2} \qquad (2.6)$$

La **rectangularidad** de una región es otro parámetro adimensional que viene definido por el cociente entre el área de la región y el área de su rectángulo base,

$$r = \frac{A}{ab} \qquad (2.7)$$

siendo *a* la longitud del lado mayor de su rectángulo base y *b* la longitud de su lado menor.

El **alargamiento** de una región se puede definir por el cociente entre la longitud del lado mayor *a* y el lado menor *b* de su rectángulo base. Sin embargo, para regiones curvadas no es una medida adecuada (Figura 2.4). Por ello, se define el alargamiento como el cociente entre su área y el cuadrado del valor máximo de su grosura,

$$Alargamiento = \frac{A}{(2d)^2} \qquad (2.8)$$

donde *d* es el número de veces que hay que aplicar el operador erosión[1] 3x3 hasta que la región desaparece.

[1] Operador erosión, página 275 de [1].

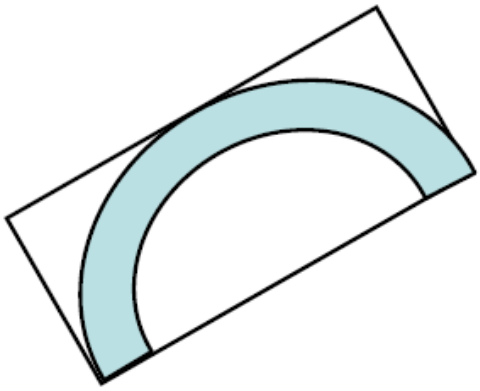


Figura 2.4. Rectángulo base de un objeto. (*Ref. [9]*).

De todos estos parámetros se han usado de forma directa, dentro del vector de características, la compacidad y los relacionados con la razón de aspecto, como la compacidad, la rectangularidad o la tasa circular.

2.1.3.3 Irregularidades

Dentro de este apartado vamos a explicar la envoltura convexa, más conocida como "convex hull" en terminología anglosajona.

Un conjunto convexo se define como un conjunto que contiene cada segmento de línea que conecta dos de sus puntos.

La **envoltura convexa** *H* de un conjunto arbitrario *S*, es el conjunto convexo más pequeño conteniendo *S*. Si *S* es convexo se cumple que *H* = *S*. Si *S* tiene solamente una componente convexa, entonces *H* puede verse como el conjunto encerrado por una goma elástica alrededor del perímetro *S*. En la figura siguiente, podemos observar una figura y su envoltura convexa.

(a) (b)

Figura 2.5. (a) Figura original y (b) su envoltura convexa (*Ref. [1]*).

Por otro lado, también se puede definir la **deficiencia convexa** *D* como el conjunto diferencia *H - S*. Se muestra en la siguiente figura.

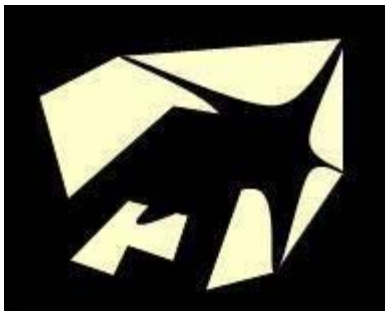


Figura 2.6. Deficiencia convexa (*Ref. [1]*).

Como se observa con las dos figuras anteriores, cualquier conjunto queda completamente definido por su envoltura convexa, que es más simple que la figura misma, y su deficiencia convexa.

En la aplicación software, dentro del vector de características, se ha usado la envoltura convexa. Concretamente, el valor utilizado dentro de este vector se corresponde con,

$$ConvexHull = \frac{ConvexArea}{Area} \qquad (2.9)$$

De esta forma, los valores que tomará este parámetro serán $ConvexHull \geq 1$. El valor más eficiente para este parámetro es la aproximación a uno, indicando que el área de la región es convexa.

2.2 ALGORITMOS PARA EL POST-PROCESAMIENTO

2.2.1 Introducción

Después de hablar sobre parámetros para la descripción de regiones, pasamos a explicar diferentes formas de realizar el post-procesamiento.

El objetivo principal del post-procesamiento es reducir la sobre-segmentación, separando la imagen en regiones grandes con significado evitando las regiones pequeñas, lo cual es muy importante ya que aumenta la eficiencia para la búsqueda de imágenes por contenido.

Una vez que se ha realizado la segmentación y, por consiguiente, la división de la imagen en regiones, se procede a localizar las regiones fragmentadas en regiones muy pequeñas. De alguna forma se calcula el número de regiones adyacentes entre una región fragmentada y las regiones más grandes. Se fusionan los fragmentos en una región mayor. Todos los pasos anteriores se repiten hasta que las regiones dejan de estar fragmentadas.

Se pueden diferenciar dos grandes bloques en el post-procesado. Por un lado, el proceso del crecimiento de regiones y, por el otro, el bloque correspondiente a la fusión de algunas regiones.

Conviene explicar que, a menudo, el post-procesamiento va unido a la segmentación realizada. Esto se debe a que algunas técnicas de post-procesado funcionan mejor con un algoritmo de segmentación que con otro, debido a que se puede obtener cierta información de la segmentación para el posterior post-procesamiento.

Para comparar y agrupar regiones se necesitan realizar algunos cálculos y desarrollar algoritmos para realizar esta tarea de la forma más automática posible. Existen multitud de formas de realizar el post-procesado de una imagen. A continuación, pasamos a explicar algunas de estas técnicas.

Toda la documentación para la realización de este apartado se ha obtenido en [12], [13], [14], [15], [16], [17] y [18].

2.2.2 Algoritmos

Vamos a dividir las diferentes técnicas que existen para realizar el post-procesado en imágenes en función del tipo de parámetros que se usan para la decisión en la fusión de regiones.

2.2.2.1 Basados en umbral

Algoritmo 1:

La técnica de post-procesado de imágenes desarrollada en [12] está basada en la información que se obtiene del matiz y de la saturación. Antes de realizar este algoritmo se pasa la imagen al espacio *HSI* (*Hue-Saturation-Intensity*) y se realiza la segmentación de la imagen con el algoritmo watershed.

Antes de realizar el crecimiento de regiones, se construye una matriz $N_x(N+2)$, siendo N el número de regiones. Esta matriz almacena en cada fila las regiones vecinas de una región, junto con el número total de sus vecinos y con una etiqueta que denota qué región semilla será combinada.

El proceso de crecimiento de regiones sigue los siguientes pasos:

En primer lugar, se realiza el proceso de selección de semilla automático. Este proceso usa regiones como semilla, en vez de píxeles como se hace tradicionalmente, esto se debe a que después de realizar la segmentación la imagen queda dividida en regiones. Para la selección automática de semillas, tienen que cumplirse tres criterios, la semilla debe ser muy similar a sus vecinos; segundo, para una región esperada, al menos ha de generarse una semilla para producir dicha región; y, finalmente, las semillas de las distintas regiones no han de estar conectadas.

Después de tener localizadas las semillas de cada región, se procede a etiquetar cada una con su propio número de secuencia.

Se recorre con un bucle cada fila de la matriz y, con otro bucle, su tabla de regiones vecinas, hasta que todas regiones estén etiquetadas. Se saca al vecino

de la región semilla, denotándolo con *p*, éste no ha de ser una región semilla. Si todos los vecinos etiquetados de *p* tienen la misma etiqueta, se fija *p* con esa etiqueta. Si no, se calcula la diferencia del valor de la media del matiz (*Hue*) de *p* con todas las regiones vecinas y se clasifica *p* con la región con la que tenga la diferencia más pequeña.

Después de etiquetar *p*, hay que borrarla de la tabla de las regiones vecinas de la región semilla y, por tanto, se resta 1 al número de regiones vecinas de esta región. La región fusionada con *p* se denota como *q*. Después de la fusión de *p*, primero se actualiza el número total de píxeles, el valor de la media del matiz y de la media de la saturación de la región *q*. A continuación, también se actualiza las regiones vecinas de *q*, añadiendo todas las regiones vecinas de *p* que no están etiquetadas. Finalmente, se actualiza el número de regiones de todas las regiones vecinas de la región *q*.

Por último, se procede a realizar la fusión de regiones. Existe la posibilidad de que algunas regiones semilla se dividan en regiones más pequeñas, así que, para evitar el problema de la sobre-segmentación, se introducen dos reglas para el fusionado de regiones.

Regla número 1. Dados dos regiones vecinas cualesquiera, se fusionan en una, si el valor de la diferencia del matiz y de la saturación de ambas es menor que un umbral dado. Según las pruebas realizadas en [12], estos valores se corresponden a 0.02 para el umbral de la diferencia de media del matiz y 0.03 para la diferencia de medias de la saturación.

Regla número 2. Si el número de píxeles en una región es menor que cierto umbral, se fusiona esta región con el vecino con el que menor sea la diferencia de color. Este umbral se corresponde con 1/150 según experimentos de [12]. Este proceso se repite hasta que ninguna región contenga menos píxeles que el umbral dado.

Podemos ver el resultado de este proceso en la siguiente figura.

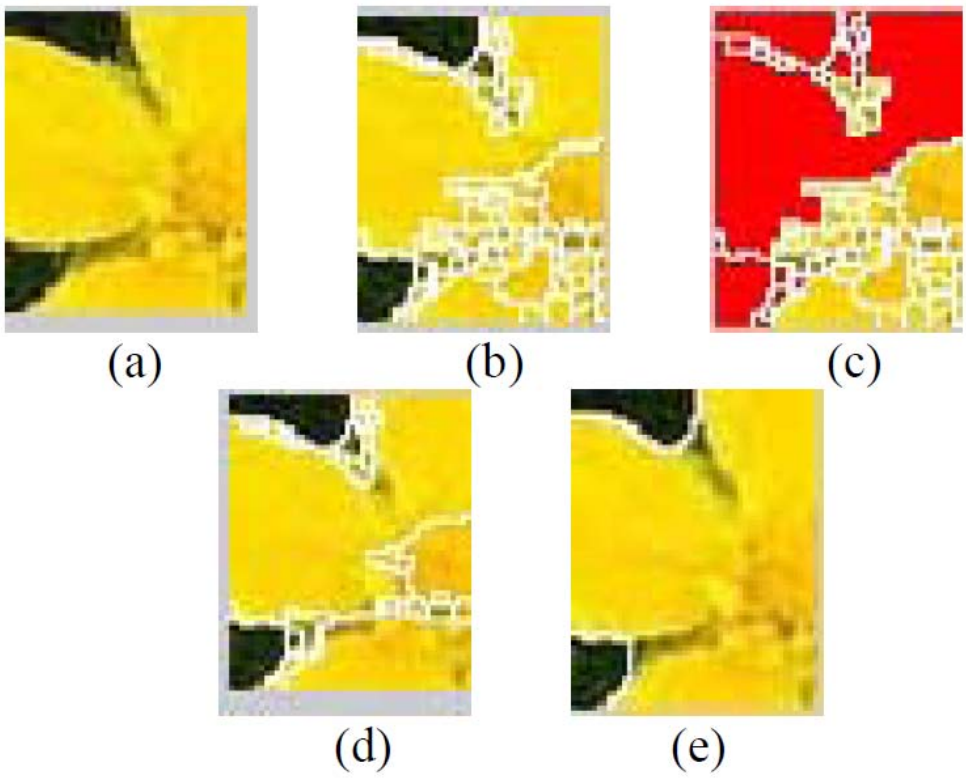


Figura 2.7. Resultado del algoritmo. (a) Imagen original. (b) Resultado de realizar la segmentación mediante watershed. (c) Se muestran las regiones semillas en rojo. (d) Resultado del crecimiento de regiones semilla. (e) Resultado final después de realizar la fusión de regiones (*Ref. [12]*).

Algoritmo 2:

Otro algoritmo de post-procesado realizado en [15], muy parecido al anterior, para evitar las regiones pequeñas debidas a la sobre-segmentación aplica dos criterios para la fusión de regiones.

Uno se basa en la similitud y otro en el tamaño. Si la diferencia de la media de color entre dos regiones vecinas es menor que un cierto umbral, se fusionan las dos regiones en una. Desafortunadamente, el resultado de la fusión de regiones depende en el orden en que se examinan las regiones. En el método que vemos en [15], primero examinan de forma automática las dos regiones que tienen la distancia menor en comparación con el resto.

En cada iteración, si el valor de la distancia entre dos regiones es menor que un cierto umbral, se fusionan estas dos regiones en una y, se vuelve a calcular la media de la nueva región, así como las nuevas distancias entre la nueva región y

sus vecinos. Se repite este proceso hasta que no existe ninguna región con la distancia menor a ese umbral dado.

La diferencia de color entre dos regiones adyacentes, R_i y R_j, se define mediante la distancia Euclidea:

$$d(R_i, R_j) = \frac{\sqrt{\left(\overline{Y_i} - \overline{Y_j}\right)^2 + \left(\overline{C_{b_i}} - \overline{C_{b_j}}\right)^2 + \left(\overline{C_{r_i}} - \overline{C_{r_j}}\right)^2}}{\min\left(\sqrt{\overline{Y_i}^2 + \overline{C_{b_i}}^2 + \overline{C_{r_i}}^2}, \sqrt{\overline{Y_j}^2 + \overline{C_{b_j}}^2 + \overline{C_{r_j}}^2}\right)} \qquad (2.10)$$

Según las pruebas realizadas en [15], se selecciona 0.1 como el umbral de similitud de color.

Después, comprobamos el tamaño de la región. Si el número de píxeles en una región es menor que un umbral, se fusiona esta región con la región vecina con la que tenga menor diferencia de color. Este procedimiento se repite hasta que no haya regiones que contengan un número menor de píxeles que el umbral dado. Basándonos en [15], este umbral se corresponde con 1/150.

Vemos el resultado de aplicar los pasos de este algoritmo en la siguiente figura.

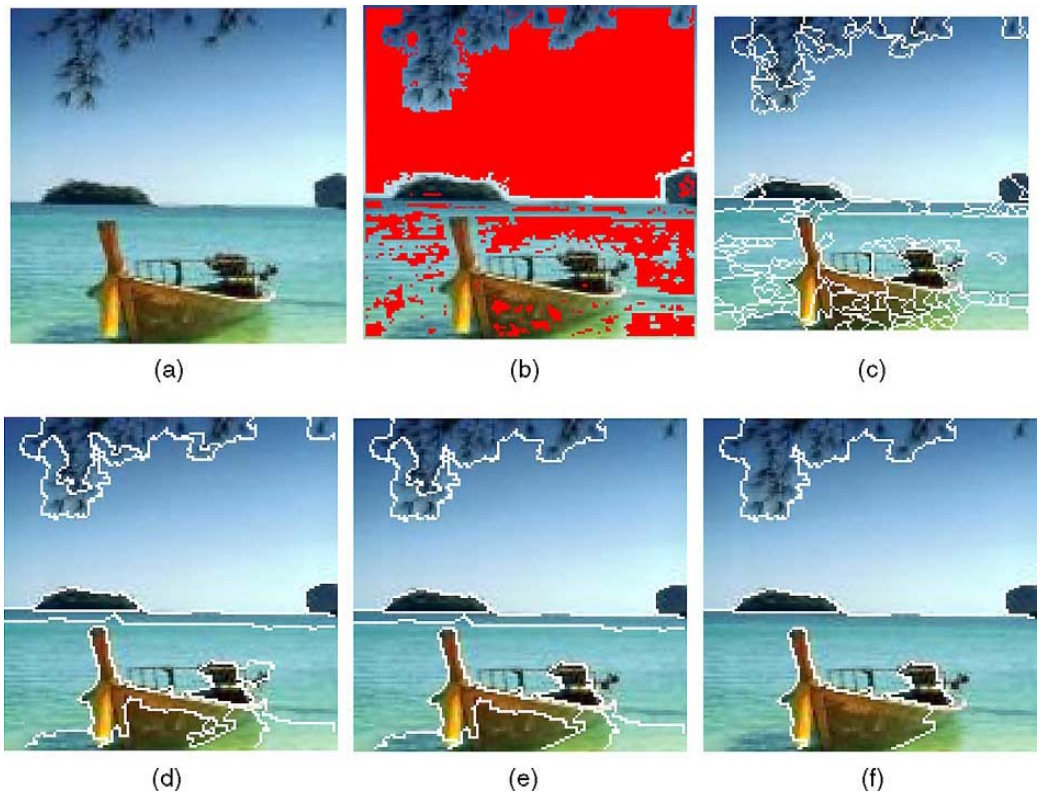


Figura 2.8. (a) Imagen original. (b) Imagen con las semillas en rojo. (c) Resultado del proceso de crecimiento de regiones. (d) Imagen resultado de fusionar regiones vecinas con la distancia Euclidea menor que 0.1. (e) Imagen resultado de fusionar regiones pequeñas con un número de píxeles menor que 1/150. (f) Resultado final *(Ref. [15]).*

Puesto que cada imagen tiene un grado de complejidad diferente, para alcanzar un post-procesado óptimo se puede variar el valor de los parámetros descritos anteriormente. En el ejemplo anterior, el cielo y el mar no se han fusionado mientras que, si se usará un valor del umbral de 0.15 estas dos regiones se fusionarían en una por similitud.

Otra información que ayudaría a mejorar el post-procesado es conocer a priori el número de regiones finales en una imagen pero, sin embargo, esta información es muy difícil conocerla.

2.2.2.2 Basados en factor de reducción

Este algoritmo de post-procesamiento se ejecuta después de realizar la segmentación watershed sobre la imagen original.[2]

En primer lugar, se empieza etiquetando cada componente conectada de la imagen. Se continúa calculando, para cada región etiquetada, el área de la región (*A*) y el número (*N*) de sub-regiones generadas por el algoritmo watershed que contiene la región.

Con estos dos parámetros se calcula un factor (*F*) que los relaciona,

$$F = \frac{A}{N} \qquad (2.11)$$

Este factor refleja el grado de sobre-segmentación de la imagen. Los valores de *F* muy altos indican la presencia de regiones innecesarias. Por tanto, el factor de reducción de regiones es proporcional a *F*. Cuanto mayor es *F* mayor será reducción que sufrirá la región correspondiente. En el siguiente paso del proceso, se reduce el tamaño de cada región.

Este factor no se puede fijar a priori, no todas las regiones deben ser reducidas de igual modo. Hay que tener en cuenta el grado de sobre-segmentación que presentan.

Después se aplica, nuevamente, el algoritmo de watershed sobre cada región reducida. Así, se obtiene un resultado mejor puesto que el grado de sobre-segmentación ha disminuido ya que ha habido una pérdida de información.

Cada región reducida, se re-escala a su tamaño original. La nueva región re-escalada tiene aproximadamente la misma forma que tenía originalmente, pero con un número menor de subregiones.

La fusión de sub-regiones se realiza por medio de la comparación de sub-regiones de la región original con la re-escalada. Las sub-regiones de la imagen original se fusionan hasta que se alcanza el mayor grado de similitud a las sub-

[2] Versión extendida del algoritmo en [13].

regiones de la imagen re-escalada. De este modo, al trabajar finalmente sobre la región original, no estamos perdiendo información sobre la forma exacta de la región global.

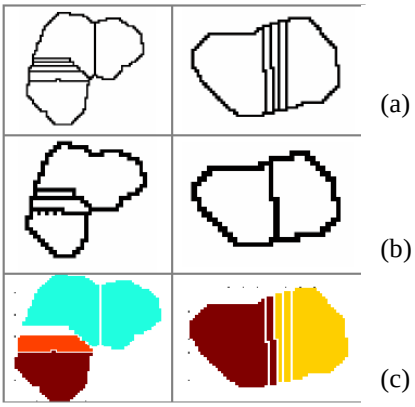


Figura 2.9. (a) Imagenes después de aplicar el algoritmo de watershed. (b) Regiones dividas en sub-regiones después de la segunda aplicación de watershed sobre las regiones reducidas. (c) Resultado final. *(Ref. [13]).*

El procedimiento se repite para cada región etiquetada.

Este método es muy bueno en imágenes con núcleos solapados. Podemos ver un ejemplo en la figura siguiente.

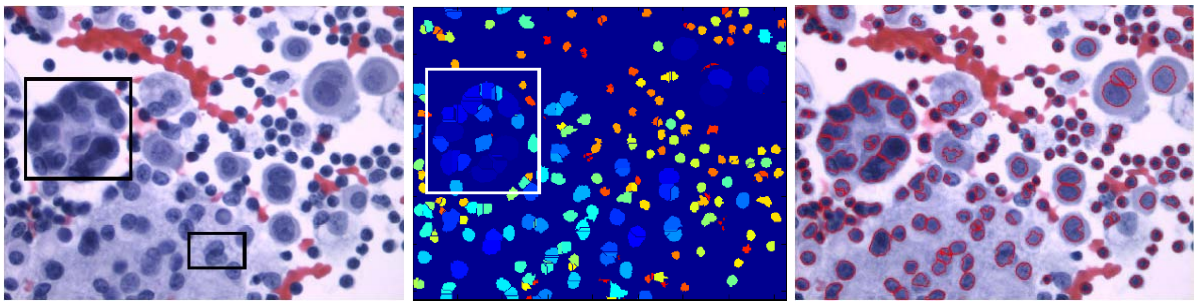


Figura 2.10. (a) Imagen original con núcleos solapados. (b) Resultado de la aplicación de la transformada watershed. (c) Resultado final aplicando el algoritmo de fusión de regiones (*Ref. [13]).*

2.2.2.3 Fusionado jerárquico

Una vez realizada la segmentación, en este tipo de algoritmo se propone calcular la disimilitud entre todo par de regiones adyacentes.

En el fusionado jerárquico de regiones, la secuencia buscada de fusión se construye paso a paso, donde en cada paso se unen un par de regiones de acuerdo con el mínimo valor de una función *F*.

Fusión basada en homogeneidad de regiones:

Esta función se trata como una función de coste, que describe la disimilitud entre regiones.

El objetivo de esta función de coste es el error cuadrático de una aproximación constante de trocitos de la imagen observada *X*, que produce una medida de aproximación exacta y se define sobre un espacio de regiones.

Consideramos que $\Re_M = \left\{R_M^1,\ R_M^2,\ \ldots ,R_M^M\right\}$ es una *M*-partición de la imagen *X* y $R_M^k = \left\{p_{k,1},\ p_{k,2},\ \ldots ,\ p_{k,\|R_M\|}\right\}$ es un conjunto de píxeles pertenecientes a la región R_M^k. En la aproximación constante de los trozos de *X*, la intensidad de la imagen en cada región R_M^k, *k = 1, 2, ..., M*, del conjunto $\Re_M$ se aproxima por un parámetro que minimiza el error cuadrático medio con respecto a los datos *X* y es igual a la media de *X* en R_M^k, que se calcula como

$$\mu\left(R_M^k\right) = \frac{1}{\left\|R_M^k\right\|}\sum_{i=1}^{\left\|R_M^k\right\|} X\left(p_{k,i}\right) \qquad (2.12)$$

donde $\|R\|$ denota la cardinalidad del conjunto *R*. El correspondiente error cuadrático es,

$$E\left(R_M^k\right) = \sum_{i=1}^{\left\|R_M^k\right\|}\left[X\left(p_{k,i}\right) - \mu\left(R_M^k\right)\right]^2 \qquad (2.13)$$

Por tanto, el total del error cuadrático es

$$E\left(\Re_M\right)=\sum_{k=1}^{M}E\left(R_M^k\right) \qquad (2.14)$$

Es evidente que, si R_M^* es la *M*-partición óptima con respecto al error cuadrático, entonces la *(M-1)*-partición óptima se genera fusionando el par de regiones R_M^* que minimiza la siguiente función de disimilitud,

$$\delta\left(R_M^{*i},R_M^{*j}\right)=\frac{\left\|R_M^{*i}\right\|\cdot\left\|R_M^{*j}\right\|}{\left\|R_M^{*i}\right\|+\left\|R_M^{*j}\right\|}\left[\mu\left(R_M^{*i}\right)-\mu\left(R_M^{*j}\right)\right]^2\mathrm{I}\left(i,j\right) \qquad (2.15)$$

donde $$\mathrm{I}\left(i,j\right)=\begin{cases}1 & \text{si las regiones } R_M^{*i},\ R_M^{*j} \text{ son adyacentes}\\ +\infty & \text{resto}\end{cases}$$

De acuerdo con toda esta formulación, el par de regiones más similar es el que minimiza (2.15).

La decisión del número óptimo de segmentos K^* se realiza comprobando el valor de $\delta\left(\cdot,\cdot\right)$. Si δ es mayor que un cierto umbral, entonces el proceso de fusión se termina. Este umbral puede ser determinado usando los conocimientos sobre la distribución del ruido.

Fusión basada en integridad de bordes:

El criterio de integridad de bordes se diseña para tener en cuenta el nivel de gradiente entre la frontera de separación de dos regiones. El coste asociado a los bordes se basa en la fracción de píxeles dentro de esta frontera.

Se define la frontera $\varepsilon_B^{i,j}$, entre dos regiones R_K^i y R_K^j, como el conjunto que contiene todos los píxeles adyacentes posibles a lo largo de la frontera común. Si el nivel de bordes de dos píxeles adyacentes es mayor que un umbral (T_h^e), entonces

estos se marcan como *píxeles duros*. El nivel de borde se define como el máximo de dos valores GMI[3] de píxeles adyacentes.

Se pueden emplear diversos criterios para la elección del umbral T_h^e. Un umbral simple y efectivo lo da la media de todos los niveles de borde.

Se define el conjunto *frontera duro* $\varepsilon_S^{i,j}$ como el conjunto de *píxeles duros* adyacentes contenidos en una imagen.

Si $\Re_K^*$ es la óptima *K*-partición entonces se obtiene la óptima *(K-1)*-partición fusionando un par de regiones de $\Re_K^*$, que minimiza la función de disimilitud siguiente,

$$\delta\left(R_K^{*i}, R_K^{*j}\right) = \frac{\left\|\varepsilon_S^{i,j}\right\|}{\left\|\varepsilon_B^{i,j}\right\|} \qquad (2.16)$$

donde $\varepsilon_S^{i,j}$ y $\varepsilon_B^{i,j}$ se definen para dos regiones adyacentes R_K^{*i} y R_K^{*j}.

Fusión basada en homogeneidad de regiones y en integridad de bordes:

Una mezcla de las técnicas anteriores se describe en [18]. Esta técnica define las matrices de adyacencia de homogeneidad, D^H, y de integridad de bordes, D^E, como,

$$\left(d_{i,j}^H\right) = \begin{cases} \delta^H\left(R_K^i, R_K^j\right) & \text{para } R_K^i,\ R_K^i \text{ adyacentes} \\ & \text{y } 1 \le i < j \le K \\ +\infty & \text{resto} \end{cases} \qquad (2.17)$$

$$\left(d_{i,j}^E\right) = \begin{cases} \delta^E\left(R_K^i, R_K^j\right) & \text{para } R_K^i,\ R_K^i \text{ adyacentes} \\ & \text{y } 1 \le i < j \le K \\ +\infty & \text{resto} \end{cases} \qquad (2.18)$$

[3] El GMI (*Gradient Magnitude Image*) es el valor del gradiente de la imagen. Ver [18].

donde $\delta^H\left(R_K^i, R_K^j\right)$ y $\delta^E\left(R_K^i, R_K^j\right)$ son las funciones de disimilitud definidas anteriormente en (2.15) y (2.16), respectivamente.

Los valores de la función de disimilitud de homogeneidad difieren en escala de los valores de integridad de bordes y su estadística debe, por tanto, ser normalizada para poderse combinar. La escala de las funciones de disimilitud se desconoce y, por tanto, se adopta un proceso de clasificación por importancia. Los elementos de D^H pueden ser ordenados por importancia para producir $d_{(1)}^H \le \ldots \le d_{(n)}^H \le \ldots \le d_{(KxK)}^H$, donde $d_{(n)}^H \leftrightarrow d_{i,j}^H$ para algún $1 \le i < j \le K$. Se define la matriz de homogeneidad ordenada $\Re^H$ como $\left(r_{i,j}^H\right) = n$, donde $d_{(r_{i,j}^H)}^H \equiv d_{i,j}^H$. Análogamente, los elementos de D^E pueden ser clasificados por orden de importancia, definiendo la matriz ordenada de integridad de bordes $\Re^E$ como $\left(r_{i,j}^E\right) = m$, donde $d_{(r_{i,j}^E)}^E \equiv d_{i,j}^E$.

Una vez que las matrices de adyacencia están ordenadas, se puede definir la matriz de pesos, W, en términos de $\Re^H$ y $\Re^E$ como,

$$W = \alpha \cdot \Re^H + (1-\alpha) \cdot \Re^E \quad 0 \le \alpha \le 1 \qquad (2.19)$$

donde α es un parámetro de ponderación. De este modo, cuando α=1 el proceso de fusión sólo se basa en el criterio de homogeneidad mientras que α=0, si únicamente se basa en la integridad de bordes. Además, para 0 < α < 1, el proceso de fusión se basa en ambos criterios y se fusionan el par de regiones que tengan el menor peso. Así, en todos los casos, el proceso de fusión une el par de regiones R_K^i y R_K^j que cumpla $w_{i,j} = \min\{W\}$.

Una vez calculados las funciones de disimilitud se procede a realizar la fusión de imágenes, para ello se usa el grafo de adyacencia de regiones, en inglés *Region Adyacency Graph (RAG)*.

El RAG es la estructura de datos utilizada para representar las regiones de una imagen segmentada. El RAG de una imagen con *K* regiones es un grafo *G=(V,E)*, donde *V = {1, ..., K}* es un conjunto de nodos y $E \subset V \times V$ es el conjunto de

segmentos que une pares de regiones. Cada región se representa con un nodo y entre dos regiones con etiquetas $i, j \in V$, si las regiones son adyacentes. En la figura que sigue podemos ver el RAG de una imagen con seis regiones.

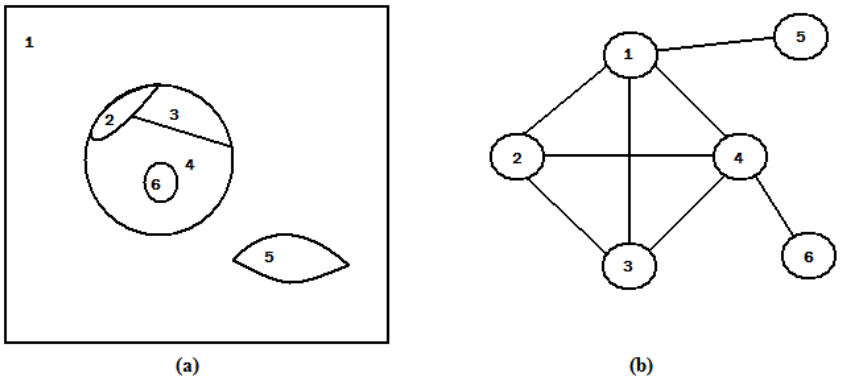


Figura 2.11. (a) Imagen con seis regiones y (b) su correspondiente RAG.

Se asigna un coste a dicho borde, en función de la disimilitud entre las dos regiones adyacentes. El par de regiones adyacentes con mayor similitud se corresponde con el mínimo coste.

Dado el RAG de una imagen con *K* regiones (*K-RAG*) se construye el RAG óptimo con (*K-n*) regiones (*(K-n)-RAG*) mediante el siguiente algoritmo iterativo:

Entrada:	RAG con *K* regiones.
Iteración:	for i = 0 : (*n-1*) Encontrar el par de regiones adyacentes con mayor similitud en el (*K-i*)-RAG. Fusionar las regiones para conseguir el *(K-i-1)*-RAG.
Salida:	RAG con (*K-n*) regiones.

Dado que, en cada paso se necesita el segmento con el coste mínimo, la estructura de datos apropiada es una cola con prioridades, que se puede implementar mediante un montículo (*heap*). Un montículo es una estructura de árbol con información perteneciente a un conjunto ordenado, donde el valor del nodo padre es menor que el valor de cualquiera de sus hijos. Por tanto, todos los segmentos del RAG están ordenados en un montículo de acuerdo con su coste.

En cada paso, el segmento con el coste mínimo se borra del montículo y los nodos que une se fusionan. Esta operación de fusión produce cambios tanto en el RAG como en el montículo. Así pues, todos los nodos cuyos vecinos hayan sido fusionados deben reestructurar su lista de vecinos, así como el coste de disimilitud con el nodo resultante de la fusión.

Podemos ver un ejemplo de fusión de dos nodos en los siguientes grafos. También se puede observar como un segmento desaparece, bien el segmento (a, e) bien el (a, b) debido a la fusión de los nodos a y b.

Figura 2.12. (a) RAG inicial y (b) RAG con fusión de los nodos *a* y *b*.

El problema de este algoritmo es que el tiempo de computación se va incrementando ya que hay que ir actualizando todos los costes de los segmentos que unen dos nodos adyacentes y, el tamaño del montículo es demasiado grande. Por lo que, para solucionar este problema existen varias versiones de este algoritmo.

Una versión descrita en [16] consiste en guardar en el montículo sólo los vecinos más semejantes, esta técnica se denomina *Most Similar Neighbor Graph* (*MSNG*). Otra versión del algoritmo es la desarrollada en [17] denominada como *Nearest Neighbor Graph* (*NNG*) la cual sólo incluye los vecinos más cercanos.

De este modo, con estas técnicas, se acelera el proceso de fusión de regiones.

2.2.3 Conclusiones

Cada uno de los algoritmos explicados anteriormente tiene un punto fuerte, por ello, para el desarrollo de los algoritmos de la herramienta desarrollada en este proyecto hemos cogido un poquito de cada uno.

Como en los primeros algoritmos, explicados en el apartado 2.2.2.1, nuestros algoritmos también se basan en un umbral para realizar la decisión de si es conveniente fusionar o no. Solamente destacar que en los algoritmos explicados se basan en una o dos características para realizar la decisión mientras que, nosotros usamos hasta cinco descriptores para la extracción de información de cada una de las regiones de una imagen. En este sentido, nuestro algoritmo es mucho más potente, ya que realiza la fusión con más información que en estos algoritmos.

Otra característica de nuestros algoritmos es que también utiliza una estructura para llevar a cabo las combinaciones posibles. Esta estructura no es un RAG tal y como se explico en el apartado 2.2.2.3. Nuestra estructura guarda información sobre todas las combinaciones que se pueden hacer con las regiones de una imagen.

Otro punto que diferencia a nuestros algoritmos es que en la estructura también se guarda fusiones de más de dos regiones, es decir, que para realizar la decisión de si es conveniente o no fusionar también cuenta con las combinaciones de tres y más regiones. Estas combinaciones sólo se comprueban en caso de que no haya ningún par de regiones con las que se obtenga un valor menor que el umbral. Esta característica se puede ver como una ventaja o un inconveniente, ventaja porque contamos con mayor número de combinaciones e inconveniente porque esto se traduce a mayor tiempo de cómputo.

En los siguientes capítulos explicaremos en profundidad los algoritmos que hemos desarrollado, de esta forma, se podrán entender mejor las líneas anteriores.

CAPÍTULO 3

DESARROLLO DEL PROYECTO

Diseño algorítmico

En este capítulo se va a explicar el diseño de los algoritmos desarrollados para la aplicación.

3.1 INTRODUCCIÓN

En este capítulo se va explicar los diferentes algoritmos desarrollados en el módulo de post-procesamiento realizado en el proyecto. El post-procesado es un proceso para mejorar la segmentación de una imagen. Una vista general de este módulo dentro del tratamiento digital de las imágenes se puede ver en la figura 3.1.

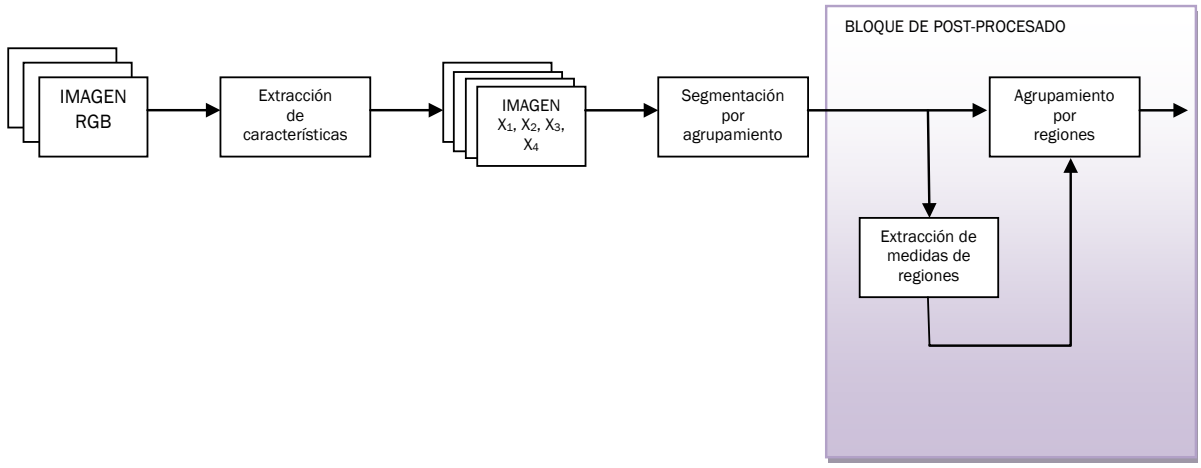


Figura 3.1. Esquema general tratamiento digital de las imágenes.

Una vez realizada la segmentación de la imagen, se realiza el post-procesamiento de ésta para reducir las regiones pequeñas, agrupándolas en regiones más significativas. Este proceso consiste en ir comparando y, si procede, agrupando regiones. La vista general de este módulo por separado podemos verla en la figura 3.2.

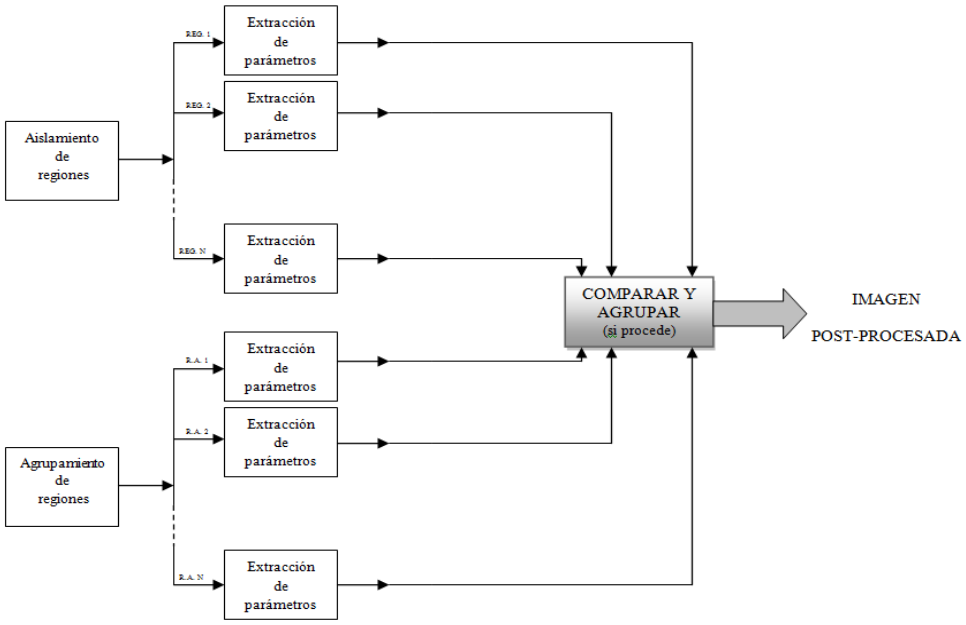


Figura 3.2. Vista general bloque post-procesamiento.

Más concretamente, una vez que se realiza la segmentación, se calcula una posible combinación de regiones de las que se van a extraer los parámetros para comparar y agrupar si así se mejora la segmentación. El diagrama de flujo con los pasos que sigue la aplicación es,

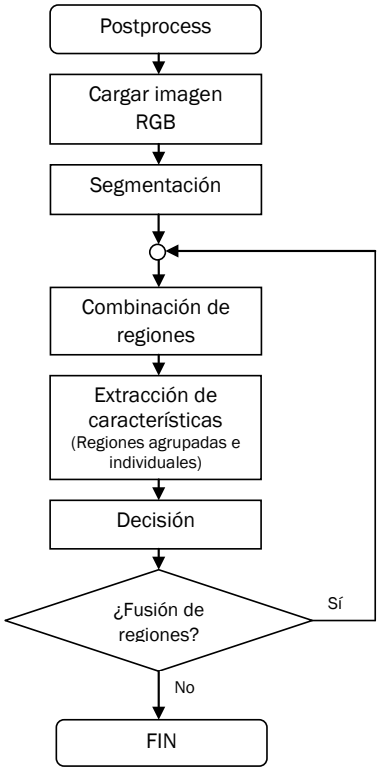


Figura 3.3. Diagrama de flujo de la herramienta software.

A pesar de que en la aplicación se realiza segmentación, no vamos a hablar de cómo se hace en esta aplicación ya que no es parte de este proyecto. Por lo tanto, vamos a pasar a explicar los algoritmos usados para la combinación de regiones, extracción de características, decisión y fusión de regiones.

3.2 COMBINACIÓN DE REGIONES

En este módulo de la herramienta, se realiza la combinación de regiones posibles para su posterior procesado.

En primer lugar, se calculan el número total de combinaciones entre regiones esto es 2^{nr}, siendo *nr* el número de regiones de la imagen segmentada. Pero, como sabemos, este número de combinaciones no es real ya que también es una posible combinación no fusionar nada o fusionar todo, lo que en nuestra aplicación carece de sentido. Otra posible combinación que nos da este cálculo es una región individual, que tampoco corresponde con ninguna fusión real. Por ello, el número de combinaciones reales para una imagen segmentada dada será $2^{nr} - nr - 2$. Aunque, para imágenes con un número de regiones elevado podemos despreciar estos casos y, por lo tanto, quedarnos con la aproximación de que el número de combinaciones de regiones es 2^{nr}.

El resultado de este algoritmo es una matriz de fusiones con ceros y unos. Cada columna de la matriz es una región y cada fila se corresponde con una posible combinación. Si en una celda hay un uno esa región pertenece a la combinación.

En este cálculo de combinaciones posibles nos podemos encontrar con una matriz de fusiones de *2* hasta (*nr-1*) regiones.

Para poder entender mejor el resultado de este algoritmo presentamos el siguiente caso. Por ejemplo, para una imagen segmentada en cuatro regiones, obtendremos una matriz de ceros y unos de tamaño $4 \times \left[2^4 - 4 - 2\right]$, es decir, de 4×10. La siguiente tabla muestra el ejemplo para una imagen de cuatro regiones.

Tabla 3.1. Matriz de fusiones para una imagen segmentada en cuatro regiones.

Región	1	2	3	4
1	0	1	0	1
2	1	1	0	1
3	1	0	1	0
4	0	1	1	1
5	1	0	0	1
6	1	1	0	0
7	0	0	1	1
8	1	0	1	1
9	0	1	1	0
10	1	1	1	0

A continuación, para cada una de las combinaciones de la matriz, se calculará la imagen binarizada resultante de cada combinación para así poder calcular el vector de características de cada una de ellas y, posteriormente, realizar la decisión de si es buena o no dicha fusión.

Se puede observar que según el número de regiones de la imagen segmentada, el resultado del número de combinaciones posibles de este algoritmo será mayor lo que afectará, exponencialmente, al tiempo de cómputo. El tiempo de cómputo se ve incrementado ya que, después, se calcularán los parámetros para todas las combinaciones. En la figura 3.4, podemos ver el aumento exponencial de combinaciones y, por tanto, de tiempo.

Tabla 3.2. Tabla con el número de combinaciones en función del número de regiones.

Número de regiones	Combinaciones posibles
3	3
4	10
5	25
6	56
7	119
8	246
9	501
10	1012
11	2035
12	4082

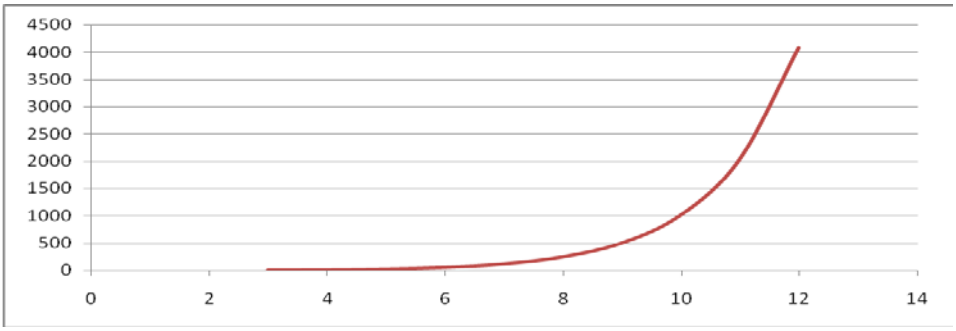


Figura 3.4. Gráfica de la tabla 3.2.

3.3 EXTRACCIÓN DE CARACERÍSTICAS

En este módulo se realiza la extracción de características sobre las regiones individuales así como sobre las regiones que han sido agrupadas en el módulo anterior.

La extracción de características consiste en calcular los parámetros necesarios para realizar la decisión de la fusión de regiones posterior. En este proyecto, principalmente se usan parámetros basados en la elongación o razón de aspecto de la región. Esto significa que, por ejemplo, en el caso de un rectángulo es la razón de su longitud a su ancho. Se han utilizado este tipo de parámetros porque los objetos de la vida real casi siempre se pueden asemejar a formas geométricas como círculos, cuadrados o rectángulos. También se han utilizado parámetros que nos informan de la dispersión de una región.

Así pues, los parámetros que se pueden utilizar son las componentes conectadas, compacidad, excentricidad, convexidad o rectangularidad.

Una vez calculadas las características de una imagen segmentada, se formará el vector de características usado en la decisión.

Antes de comenzar a explicar cada uno de los parámetros, para evitar repeticiones innecesarias conviene aclarar que cada medida se realiza para una determinada región R_i perteneciente al espacio $\Re_M = \{R_1, R_2, ..., R_M\}$, que contiene todas las regiones de una imagen.

A continuación, se explica la forma de calcular cada uno de estos parámetros por separado ya que el usuario puede hacer uso de cada parámetro de forma independiente.

3.3.1 Componentes conectadas

Como se explico en esta memoria[4], una componente conectada o conexa de una imagen es un conjunto de píxeles tal que para cualquier par de píxeles del

[4] Sección 2.1.3.1.

conjunto, existe un camino que los une, de tal manera que todos los píxeles tienen al menos un vecino perteneciente a dicho conjunto.

Este parámetro se utiliza para calcular la dispersión de una región, cuanto mayor sea el número de componentes conectadas más dispersa estará la región y, por consiguiente, se necesitará realizar el post-procesado de dicha región.

La forma para calcular el número de componentes conectadas se realiza primero, binarizando cada una de las regiones (ver figura 3.5 (b)) y, a continuación, se calcula la vecindad a 8 de cada píxel. De esta forma, obtenemos el número de subregiones, componentes conectadas, de una región.

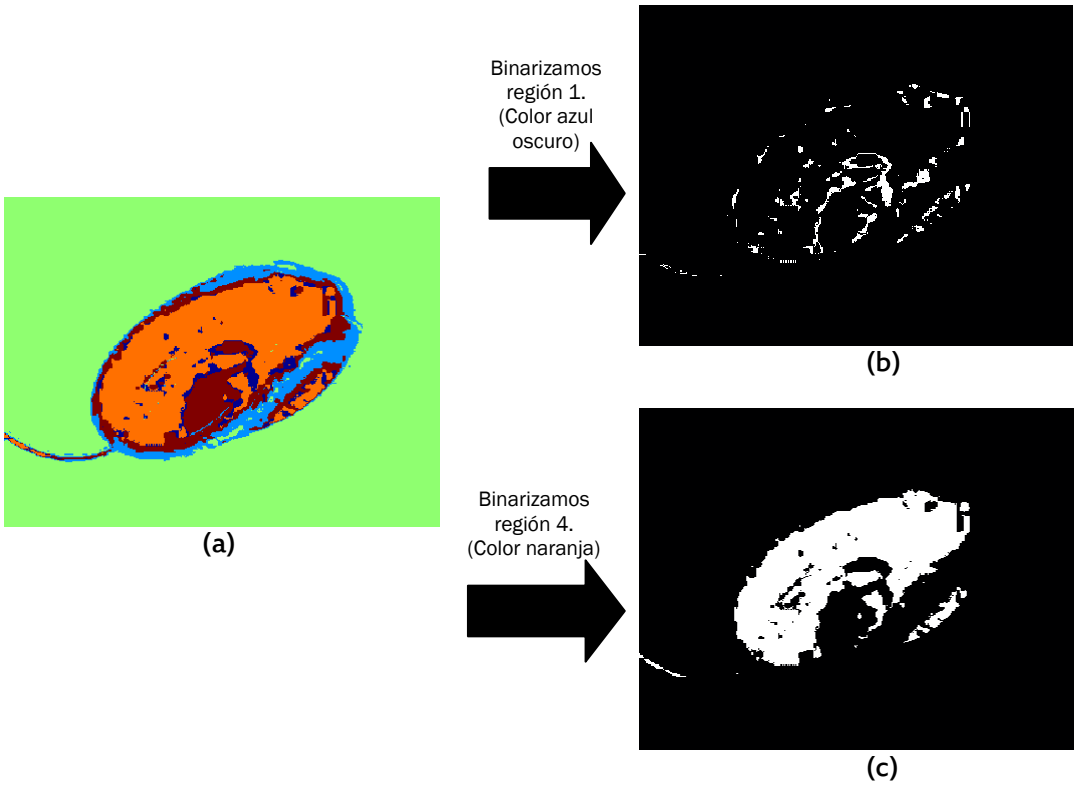


Figura 3.5. Ejemplo de cálculo de número de componentes conectadas. (a) Imagen original segmentada en 5 regiones. (b) Región 1 binarizada con 115 componentes conectadas. (c) Región 4 binarizada con 24 componentes conectadas.

Como vemos en el ejemplo de la figura 3.5, la región 1 está muy dispersa y, por ello, el número de componentes conexas es muy elevado. Por lo que será interesante fusionar esta región con otra, de tal maneta que obtengamos un menor número de componentes conectadas y, así, una región más compacta. Mientras

que en 3.5 (c) vemos que la región cuatro es más compacta y, por lo tanto, el número de componentes conectadas es menor que en (b).

3.3.2 Compacidad

La compacidad[5], también llamada circularidad, de una región es un tipo de descripción muy usado ya que relaciona el perímetro de una región con el área de ésta.

Para calcular este parámetro se ha utilizado la ecuación (2.5) de esta memoria para cada una de las regiones.

$$C(R_i) = \frac{Área(R_i)}{\left[Perímetro(R_i)\right]^2} \qquad (3.1)$$

donde *C* se refiere a la compacidad y R_i es una región de la imagen.

Si la compacidad es un valor alto esto significará que la región es compacta. Esto representa que se fusionarán aquellas regiones con las que se obtengan valores de compacidad mayores que con estas mismas regiones sin fusionar.

Por ejemplo, con las siguientes imágenes (figura 3.6) comprobamos como para la región (a) el valor de la compacidad es mayor que para la región (b), siendo que ambas regiones tienen el mismo área.

[5] Sección 2.1.3.2 de esta memoria.

(a) (b)

Figura 3.6. (a) Compacidad = 0.0724 y (b) Compacidad = 0.0033.

3.3.3 Excentricidad

La excentricidad es el parámetro que mide la relación entre el eje mayor y el eje menor de una región. La forma más simple de medirlo es calcular la relación entre la cuerda mayor respecto a la cuerda perpendicular a la cuerda mayor.

La ecuación con la que se consigue esta medida es,

$$E(R_i) = \frac{Longitud\_eje\_mayor(R_i)}{Longitud\_eje\_menor(R_i)} \qquad (3.2)$$

donde *E* es la excentricidad y R_i es una región de la imagen.

La excentricidad mide el grado de relación entre los ejes, es decir, aporta el grado de uniformidad de una región. Por lo que los valores próximos a uno de este parámetro indican una forma perfecta, es decir, un cuadrado o un círculo. A continuación, pasamos a ver varios ejemplos.

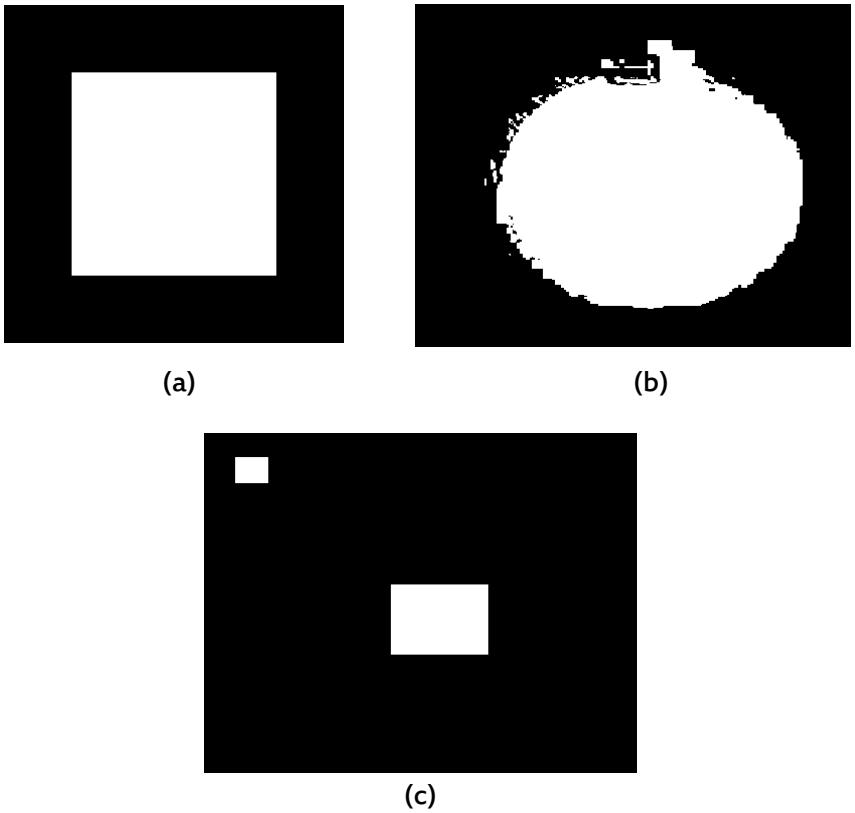


Figura 3.7. (a) Excentricidad = 1. (b) Excentricidad = 1.0288. (c) Excentricidad = 3.4316.

Se puede observar en la figura 3.7 (a) que con una forma que tiene el eje mayor igual que el eje menor se obtiene el mínimo valor de excentricidad, 1. Para la región (b) comprobamos que se alcanza un valor próximo a 1, ya que a pesar de no tener los ejes iguales es bastante próxima a un círculo. Mientras que, para la región (c) el valor obtenido de la excentricidad es mucho mayor que en el resto de casos, esto se debe a que la región está dispersa.

Con todo esto, queda demostrado que para que dos regiones se lleguen a fusionar esta medida ha de alcanzar un valor próximo a 1.

3.3.4 Convexidad

Un conjunto convexo[6], como se definió en esta memoria, es el conjunto que contiene cada segmento de línea que conecta dos de sus puntos.

[6] Sección 2.1.3.3.1.

Generalmente, una región es convexa si y sólo si para cualquier pareja de puntos de la región, la línea recta que los une pertenece al interior de la región. La envoltura convexa de una región *R* es la menor región convexa *H* que verifique que *R* es un subconjunto de *H*.

La convexidad se calcula mediante,

$$Cx(R_i) = \frac{Área\_envoltura\_convexa(R_i)}{Área(R_i)} \qquad (3.3)$$

donde *Cx* se refiere a la convexidad y R_i de la imagen.

Este parámetro mide las formas irregulares de una región, por lo tanto, hace una buena estimación para figuras que no sean circulares o cuadradas. Para la fusión, se buscan los valores más pequeños de esta medida.

Siguiendo con el ejemplo anterior de la figura 3.7, para las tres regiones se obtienen los valores de convexidad = 1 para la región (a), convexidad = 1.1295 para (b) y convexidad = 3.016 para la región (c). El razonamiento de estos valores es el mismo que en el caso del parámetro anterior.

3.3.5 Rectangularidad

El rectángulo base[7] es el rectángulo más pequeño que contiene completamente la región.

Para calcular el rectángulo base de una región existen diversas técnicas que podemos encontrar en [19]. La técnica utilizada en este proyecto es una de las más simples. Consiste en obtener las coordenadas de la esquina más cercana a la parte superior izquierda de la imagen, [*x_min, y_min*] y las de la esquina más alejada, [*x_max, y_max*]. Una vez halladas, se restan las coordenadas [(*x_max - x_min*), (y*_max - y_min*)] y se obtienen los valores de la base y la altura del rectángulo base que contiene la región.

[7] Apartado 2.1.2.1.

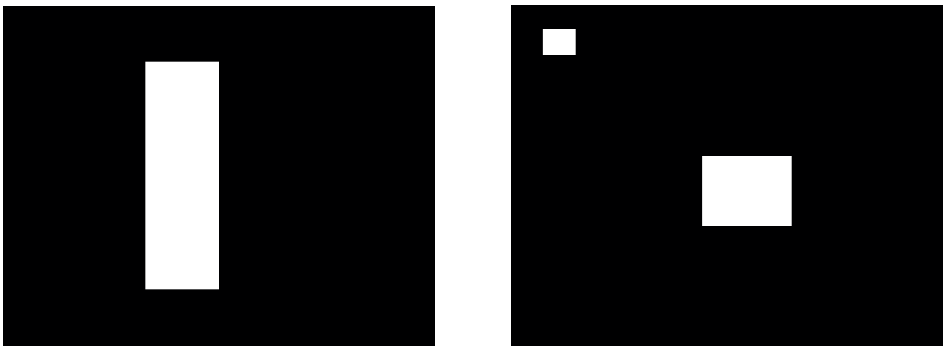
Como medida de este parámetro, dentro del vector de características, se utiliza la relación entre el área de una región respecto al área del rectángulo base, llamado rectangularidad. La ecuación que rige es,

$$R(R_i) = \frac{Área(R_i)}{Área\_rectángulo\_base(R_i)} \qquad (3.4)$$

donde R denota la rectangularidad y R_i se corresponde con una región de la imagen.

La rectangularidad también es otro parámetro que mide la relación de aspecto, su valor oscila entre [0, 1]. En este caso, los valores próximos a 1 se corresponden con formas cuadradas perfectas.

En la siguiente figura vemos el ejemplo en el que con una región perfectamente rectangular (figura 3.8 (a)) obtenemos un valor de rectangularidad igual a 1 mientras que, para una región compuesta de dos rectángulos no adyacentes (figura 3.8 (b)) el valor de la rectangularidad es próximo a 0.



(a) (b)

Figura 3.8. (a) Rectangularidad = 1. (b) Rectangularidad = 0.14611.

3.3.6 Tasa circular

La tasa circular es otro parámetro que se uso en el vector de características, que como se explicó anteriormente[8], se corresponde con un parámetro normalizado muy similar a la compacidad de una región.

El valor de la tasa circular se calcula con la siguiente ecuación,

$$R_c(R_i) = 4\pi \frac{Área(R_i)}{\left[Perímetro(R_i)\right]^2} \qquad (3.5)$$

donde R_c se refiere a la tasa circular y R_i es una región contenida en la imagen segmentada.

El valor de esta medida es 1 para regiones circulares y $\pi / 4$ para regiones cuadradas.

Puesto que esta medida está bastante relacionada con la compacidad no nos aporta información en la decisión, por lo que, finalmente, no se incluyó en la herramienta de post-procesado descrita en esta memoria.

3.3.7 Redondez

La redondez es otro parámetro para calcular la razón de aspecto de una región.

La redondez se obtiene mediante la relación entre el área de una región y su eje mayor al cuadrado. Se calcula con la ecuación siguiente,

$$Redondez(R_i) = \frac{Área(R_i)}{\left[Eje\_mayor(R_i)\right]^2} \qquad (3.6)$$

donde R_i es una región del espacio de regiones.

[8] Sección 2.1.3.2.

Finalmente, este parámetro también se ha eliminado del vector de características de la aplicación puesto que también es otra medida muy similar a la compacidad y no aporta información en la decisión y, por lo tanto, lo único que se pierde es eficiencia al incluirlo.

3.3.8 Normalización de parámetros

Puesto que los valores de los parámetros anteriores están muy dispersos y vamos a utilizarlos de forma conjunta para realizar la decisión, necesitamos que compartan un mismo rango de valores.

Así pues, hemos procedido a normalizarlos de forma que los valores de todos los parámetros oscilan en el rango [0, 1]. Las normalizaciones que se han hecho para cada parámetro son las siguientes.

Para las componentes conectadas, como sabemos, la situación ideal es que tome valor igual uno mientras que, deseamos que se agrupen regiones con valores mayores. Así pues, la normalización que hacemos es la siguiente,

$$\overline{cc}(R_i) = 1 - \frac{1}{cc(R_i)} \qquad (3.7)$$

Para el caso de la excentricidad, la normalización que realizamos es la misma que en el caso de las componentes conectadas. El parámetro queda así,

$$\overline{E}(R_i) = 1 - \frac{1}{E(R_i)} \qquad (3.8)$$

Como sabemos, cuando se trata de las convexidad los valores que se obtienen son mayores o iguales a uno. Siendo el caso ideal que el parámetro tome el valor igual a uno. Como ahora nos interesa que el valor mejor sea cero, la normalización que hacemos es,

$$\overline{Cx}(R_i) = 1 - \frac{1}{Cx(R_i)} \qquad (3.9)$$

Para el descriptor de la compacidad, la normalización la hacemos con ayuda de la tasa circular. El nuevo parámetro queda así,

$$\overline{C}(R_i) = 1 - \frac{4\pi Área(R_i)}{\left[Perímetro(R_i)\right]^2} \qquad (3.10)$$

El círculo es la forma más compacta que existe puesto que contiene el máxima superficie en el mínimo perímetro, por lo que el mínimo valor de este parámetro es cero.

Para la rectangularidad no hace falta hacer ningún tipo de normalización debido a que el valor de este parámetro ya está comprendido entre [0, 1] por definición.

3.3.9 Vector de características

El vector de características es el vector que contiene los valores de cada uno de los parámetros para una región dada.

$$W(R_i) = \left[\overline{cc}(R_i),\ \overline{C}(R_i),\ \overline{E}(R_i),\ \overline{Cx}(R_i),\ R(R_i)\right] \qquad (3.11)$$

donde $\overline{cc}$ es el número de componentes conectadas normalizado, $\overline{C}$ es la compacidad normalizada, $\overline{E}$ se refiere a la excentricidad normalizada, $\overline{Cx}$ a la convexidad normalizada y R a la rectangularidad. R_i es una región de la imagen segmentada.

Para facilitar los cálculos, en el bloque de decisión de la aplicación, se usa la suma de los cuadrados de la norma de las componentes del vector, esto es,

$$W_{sum}(R_i) = \sum_{i=1}^{k} |W_i|^2 \qquad (3.12)$$

donde R_i es una región del espacio de regiones de la imagen, k es el número de parámetros del vector de características de la región R_i, $W(R_i)$.

Así pues, de aquí en adelante, el valor que usaremos para realizar la decisión de si es conveniente o no fusionar una combinación de regiones es W_{sum}.

3.4 DECISIÓN Y FUSIÓN DE REGIONES

En este bloque se realiza la decisión y, si conviene, la fusión de regiones.

Una vez realizados los cálculos del vector de características de cada región, fusionada e individual, se procede a realizar la decisión. Grosso modo, esta decisión consiste en que si existe alguna combinación de regiones en la que los valores del vector de características son menores que un cierto umbral entonces, se procede al fusionado de estas regiones.

Es importante señalar que en la mayoría de los casos, con este algoritmo se obtienen dos regiones individuales que se corresponden con un objeto y un fondo.

A continuación, vamos a explicar paso por paso la ejecución del algoritmo.

En la primera iteración del bloque de decisión y fusión, se calcula el umbral que vamos a utilizar para comparar si existe alguna combinación de regiones con mejores resultados. Este umbral lo denotamos como W_{umbral}.

A continuación, pasamos a estudiar los efectos de una posible fusión. Comenzamos con la fusión de pares de regiones. Esto es, calculamos para una combinación posible de un par de regiones el valor de la suma de los vectores de características de las regiones individuales más el valor del vector de características del par de regiones que se están estudiando su fusión, W_{nuevo}. Con regiones individuales nos referimos a regiones ya formalizadas, es decir, todavía no se realiza ninguna suposición de fusión. Se trata de regiones simples o que ya han sido fusionadas en iteraciones anteriores. Matemáticamente W_{nuevo} sería,

$$W_{nuevo} = W_{sum}\left(R_{i\#j}\right) + \sum_{\substack{k=1 \\ k\neq i,j}}^{(M-2)} W_{sum}\left(R_k\right) \qquad (3.13)$$

donde $R_{i\#j}$ indica la región obtenida a partir de la fusión de las regiones *i*, *j* y R_k son el resto de regiones de la imagen.

Si el valor de W_{nuevo} es menor que el umbral W_{umbral} entonces, se modifica el valor del umbral a este nuevo valor y se estudian el resto de posibles combinaciones.

$$W_{nuevo} < W_{umbral} \Rightarrow W_{umbral} = W_{nuevo} \qquad (3.14)$$

Si es mayor, la fusión no se realiza y se pasa a estudiar otra posible combinación.

Al final de comprobar el resto de posibles combinaciones, fusionamos el par de regiones que nos han dado el mínimo valor de *W*.

Se fusiona construyendo una nueva imagen con las regiones iniciales menos una. Nuevamente, se calcularán los vectores de características para las regiones individuales y para las posibles combinaciones y, se volverá a ejecutar todo el algoritmo anterior.

Por el contrario, si no encontramos ninguna combinación de par de regiones que sea menor que el umbral inicial, pasamos a estudiar el caso de fusionar tres regiones. Repitiendo los pasos anteriores.

El umbral seleccionado para esta aplicación se calcula sumando los vectores suma, W_{sum}, de cada región individual. Matemáticamente quedaría,

$$W_{umbral} = \sum_{i=1}^{M} W_{sum}(R_i) \qquad (3.15)$$

donde *M* es el número de regiones individuales de la imagen.

También se ha probado otro umbral (3.16), mayor que el anterior exactamente, dos veces el anterior y el resultado de la decisión final es idéntico que con el umbral (3.15).

$$W_{umbral} = 2\left(\sum_{i=1}^{M} W_{sum}(R_i)\right) \qquad (3.16)$$

El fin de este algoritmo se dará cuando se haya alcanzado una imagen con dos regiones o cuando ya no exista ninguna fusión de regiones con la que se obtenga un W_{nuevo} que sea menor que el umbral.

CAPÍTULO 4

DESARROLLO DEL PROYECTO

Herramienta Software

En este capítulo explicaremos la herramienta software desarrollada.

4.1 INTRODUCCIÓN

La aplicación compuesta de todos los algoritmos explicados en el Capítulo 3 más un bloque de segmentación se ha realizado con *Matlab®*[9].

La aplicación de este proyecto está íntegramente escrita en lenguaje *M* con ayuda de la librería de procesado de imagen que incluye el software. Además, para facilitar el uso de este bloque, se ha realizado una interfaz gráfica, desarrollada también con *Matlab®*.

Como se explico en el capítulo anterior, la aplicación se compone de tres grandes bloques, uno que realiza el agrupamiento, otro donde se calculan los parámetros de las regiones individuales y de las posibles combinaciones y, otro, donde se realiza la decisión y fusión. En el siguiente diagrama de bloques podemos observar, en líneas generales, los pasos que realiza la aplicación una vez realizada la segmentación.

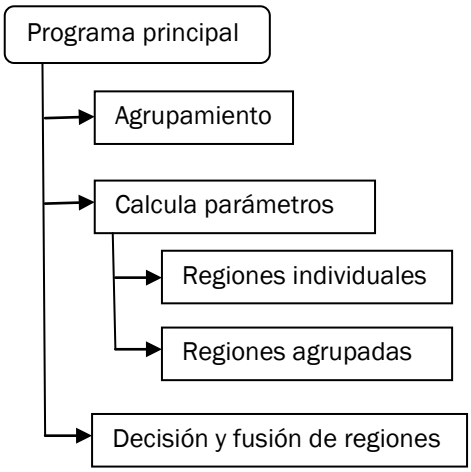


Figura 4.1. Diagrama de la aplicación principal.

[9] *Matlab®*, *MATrix LABoratory*, es un software matemático desarrollado por *The MathWorks* que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje *M*). Una característica muy importante de este software es que posee unas librerías, *toolboxes*, muy potentes. Entre ellas, se encuentra la librería de procesado de imagen.

El programa principal es un función que hemos llamado *Postprocess.m*, la llamada a esta función se realiza escribiendo en la ventana de comandos de *Matlab®* el nombre de la función o si se desea obtener o enviar algún parámetro, *Postprocess* admite un número variable de argumentos de entrada y salida. La sintaxis para la llamada a *Postprocess* es,

$$varargout = Postprocess(varargin)$$

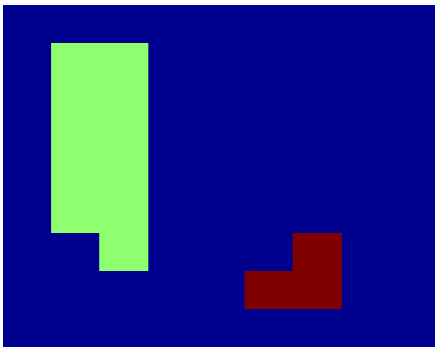
donde *varargout* puede ser cualquiera de los parámetros que la función calcula y *varargin* cualquiera de los parámetros que la función necesita. Para clarificar, por ejemplo, un parámetro de entrada puede ser la imagen original o la imagen segmentada y un parámetro de salida puede ser la imagen post-procesada.

Este capítulo se va dividir en dos apartados. En el primer apartado, contaremos los tres bloques del programa principal. Después, dedicaremos un capítulo a la interfaz gráfica desarrollada.

Para el desarrollo del software de la aplicación se han usado como referencias [4], [6] y [7]. Mientras que, para la implementación de la interfaz se han usado las referencias [5], [20], [21] y [22].

4.2 PROGRAMA PRINCIPAL

Una vez realizada la segmentación obtenemos una imagen dividida en regiones. Cada región se describe mediante una etiqueta. En la siguiente figura podemos ver un ejemplo de una imagen segmentada y su correspondiente representación mediante una matriz.



(a)

1	1	1	1	1	1	1	1	1
1	2	2	1	1	1	1	1	1
1	2	2	1	1	1	1	1	1
1	2	2	1	1	1	1	1	1
1	2	2	1	1	1	1	1	1
1	2	2	1	1	1	3	1	1
1	1	2	1	1	3	3	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

(b)

Figura 4.2. (a) Imagen segmentada y (b) su matriz de regiones.

A partir de la matriz de regiones podemos calcular todas las características de cada región. Para ello, contamos con una serie de funciones y algoritmos.

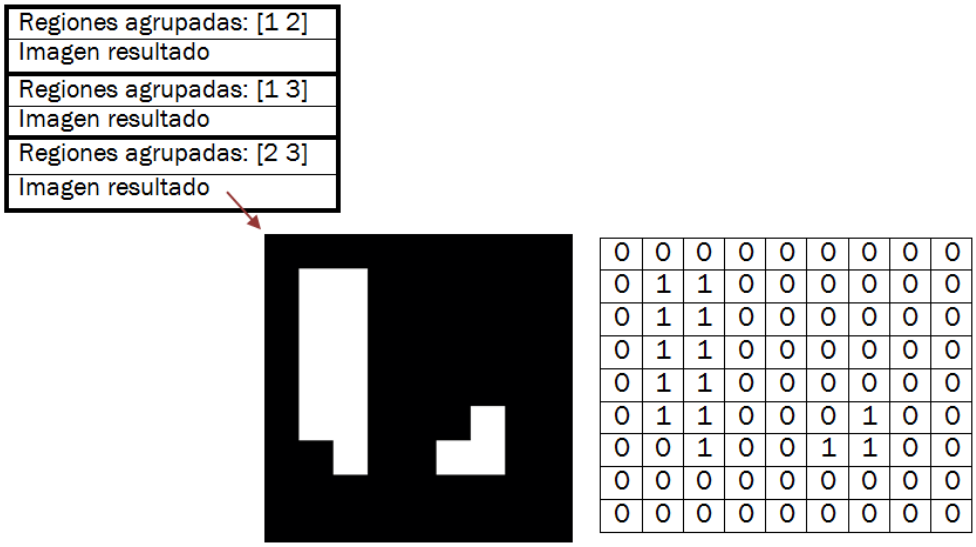
4.2.1 Bloque de agrupamiento

Lo primero que se realiza al recibir la imagen segmentada es calcular las posibles combinaciones de regiones con el algoritmo descrito en el apartado 3.2 de esta misma memoria.

Como se explico anteriormente, con el algoritmo de combinación de regiones obtenemos una matriz de fusiones que consiste en una matriz que indica que regiones se van a fusionar. A partir, de la obtención de esta matriz, calculamos una estructura que guarde en cada posición dos campos, uno que se corresponde con las regiones fusionadas y otro con la región resultante binarizada. La función encargada de realizar este algoritmo en la herramienta se denomina *agrupamiento.*m, esta funcion se llama mediante,

$$Regiones = agrupamiento(image)$$

donde *image* es la imagen segmentada de entrada que queremos post-procesar y *Regiones* es la estructura que devuelve el algoritmo. Siguiendo con el ejemplo de la figura 4.2, con este algoritmo obtendríamos una estructura de tamaño [3x1] con la información siguiente,



0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0
0	0	1	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figura 4.3. Arriba a la izquierda, resultado del algoritmo de combinaciones. Abajo izquierda, imagen resultante de la fusión de las regiones 2 y 3. Abajo derecha, matriz de regiones.

A continuación, se binariza cada región individual de la imagen y se guarda en una estructura de forma idéntica al anterior algoritmo. La función encargada de ello se llama *regionesIndividuales.m*. La función a esta llamada se realiza mediante,

$$Regiones = regionesIndividuales(image)$$

donde *image* es la imagen segmentada que queremos post-procesar y *Regiones* es la estructura que devuelve el algoritmo.

Posteriormente, se juntan estas dos estructuras para formar una sola. Así pues, la estructura final que nos servirá para el cálculo de los parámetros quedará como se indica en la figura 4.4.

Regiones agrupadas: [1]
Imagen resultado
Regiones agrupadas: [2]
Imagen resultado
Regiones agrupadas: [3]
Imagen resultado
Regiones agrupadas: [1 2]
Imagen resultado
Regiones agrupadas: [1 3]
Imagen resultado
Regiones agrupadas: [2 3]
Imagen resultado

Figura 4.4. Estructura final.

4.2.1.1 Restricciones del algoritmo

Una restricción que presenta este algoritmo es el número de combinaciones que es capaz de calcular. Como se comentó en el capítulo 3 de esta memoria, el número de combinaciones posibles se calcula en función del número de regiones y su valor es aproximadamente 2^{nr}. Pues bien, la restricción es causada por el tamaño máximo posible que soporta *Matlab®* para un array, este es de 407 MB.

Todo esto significa que el algoritmo de combinación de regiones no funciona cuando el número de regiones es mayor a 25.

4.2.2 Bloque de extracción de características

Una vez obtenida la estructura final, se procede a calcular los parámetros de cada imagen resultado hallada.

La función principal que se encarga de realizar el estudio de las características de cada región de la imagen es una función de la librería de procesado de imagen de *Matlab®*, esta función se conoce como *regionprops.m*.

Con la función *regionprops* podemos obtener medidas de las regiones como el área, el rectángulo base, el centro de masas de la región, medidas relacionadas con la envoltura convexa y los ejes mayor y menor, entre otras. Esta función

necesita como parámetro de entrada la matriz etiquetada (*L*) que es la imagen resultado y las propiedades que queremos que calcule (*properties*). La sintaxis es la siguiente,

$$STATS = regionprops(L,\ properties)$$

La función *regionprops* devuelve una estructura (*STATS*) con todas las medidas que hemos introducido como parámetro de entrada en *properties*.

Las propiedades necesarias para nuestra aplicación son:

- *Area*. El área es el número actual de píxeles en la región.
- *BoundingBox*. Esta medida se corresponde con el rectángulo base que contiene a la región. Este parámetro se describe mediante un vector *[ul_corner width]*:
    - *ul_corner* es otro vector con la forma *[x y]* que contiene las coordenadas de la esquina superior derecha.
    - *width* es un vector con la forma *[x_width y_width]* que especifica la base y la altura del rectángulo base.
- *MajorAxisLength*, que es la longitud en píxeles del eje mayor de la elipse que contiene a la región.
- *ConvexArea*. Esta medida se corresponde con el área de la envoltura convexa, que es el número de píxeles que ésta contiene.
- *Perimeter*. El perímetro es el contorno que describe a la región.

Puesto que necesitamos el perímetro para calcular algunos de los descriptores usados en la aplicación, ha sido necesario realizar una función aparte ya que la función *regionprops.m* sólo calcula el perímetro de regiones que no sean discontinuas.

Así pues, la función encargada de calcular el perímetro es *perimeter.m* cuya sintaxis es,

$$p = perimeter(region)$$

donde *p* es el valor del perímetro y *region* es la imagen con la región de la que deseamos calcular el perímetro binarizada.

Esta función hace uso de la función *bwlabel.m*, propia de la librería de procesado de imagen de *Matlab®*, cuya sintaxis es,

$$[L,\ num] = bwlabel(BW,\ n)$$

donde *BW* es la imagen con la región de interés binarizada y *n* es la vecindad, pudiendo ser de vecindad a 8 o a 4. El resultado es una imagen *L* con cada componente conectada marcada con una etiqueta y *num* es el número de componentes conectadas en la imagen.

Una vez que obtenemos *L*, calculamos el perímetro con la función *regionprops.m* de cada componente conectada. El resultado final es la suma de todos los perímetros de cada componente de la región.

Con todas estas medidas y tal como explicamos en el capítulo anterior, obtenemos el valor de los parámetros de compacidad, rectangularidad, excentricidad, convexidad y número de componentes conectadas.

A partir de aquí ya tenemos calculados todos los parámetros y podemos construir el vector de características necesario para el decisor. Este vector también se explicó en el capítulo anterior[10].

4.2.3 Bloque de decisión y fusión de regiones

Como se ha comentado en anteriores capítulos, en este bloque se trata de decidir si la fusión de algunas regiones mejora la segmentación y, si es así proceder a fusionarlas.

La realización de estos algoritmos sigue los pasos explicados en la sección 3.4 de esta memoria.

[10] Sección 3.3.8 de esta memoria.

En primer lugar, y sólo en la primera iteración del algoritmo, se calcula el umbral de referencia. Después, se van comparando con este umbral cada una de las posibles combinaciones y, si con alguna combinación se obtiene un valor del vector de características menor que el umbral, se fusiona dicha combinación.

La función encargada de realizar la fusión se denomina *newImage.m*. Esta función nos devuelve la nueva imagen resultante de la fusión de dos o más regiones. Si, por ejemplo, en una imagen segmentada en cuatro regiones mejora la segmentación la fusión de las regiones 2 y 4, todos los píxeles marcados con un cuatro pasarán a tener la etiqueta de la región dos. La sintaxis es,

$$NI = newImage(image,\ regionesFusionadas)$$

donde *image* es la imagen en la que vamos a realizar una fusion, *regionesFusionadas* son la combinación de regiones que se va a fusionar y *NI* es la nueva imagen que se obtiene.

Una vez calculada la nueva imagen, se volverá a realizar el agrupamiento, la extracción de características y la decisión. El fin del programa se dará cuando no exista ninguna combinación mejor.

4.3 INTERFAZ GRÁFICA

Para facilitar la interacción entre el usuario y el sistema se ha realizado una interfaz gráfica de usuario. De esta manera, no es necesario que el usuario tenga que tener nociones sobre la programación de la aplicación y se centre exclusivamente en la funcionalidad de la misma.

La interfaz gráfica también se ha desarrollado con *Matlab®*, ya que el software está íntegramente implementado con el lenguaje de este programa y, así evitamos problemas de interoperabilidad entre lenguajes.

Además, *Matlab®* nos ofrece un entorno de edición de interfaces de usuario (*GUIDE*) con el que es relativamente sencillo realizar el diseño de los componentes gráficos de la aplicación.

En la figura 4.5 podemos ver el aspecto visual de la interfaz desarrollada para el bloque de post-procesamiento de imágenes.

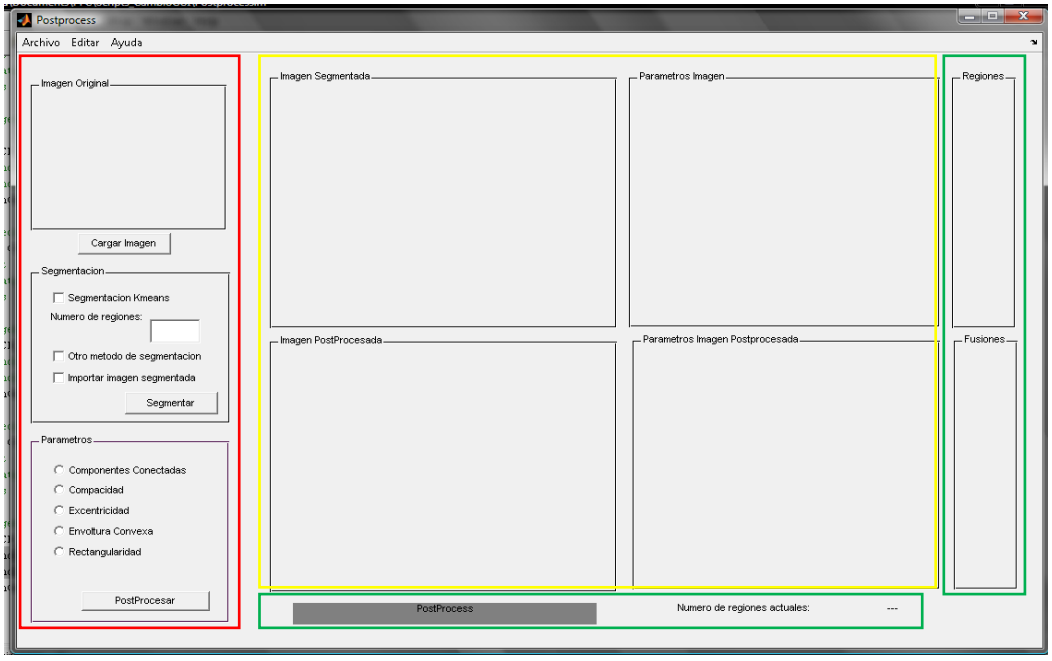


Figura 4.5. Interfaz gráfica de la aplicación.

En los paneles de la izquierda, marcados de rojo, es con los que se realiza la ejecución de los algoritmos. El panel situado más abajo dentro de esta zona (figura 4.6) es el que se encarga de realizar el post-procesado como tal. Si nos fijamos, el usuario puede hacer uso de cada parámetro por separado.

Figura 4.6. Panel de elección de parámetros y post-procesado.

En los paneles de amarillo de la figura 4.5 vemos la visualización de imágenes y de las gráficas de los parámetros. Mientras que, las zonas marcadas de verde son apoyo, es decir, de información adicional como, por ejemplo, el estado de la aplicación, el número de regiones o las fusiones que se han realizado.

Gracias al uso de la interfaz, el usuario puede ver paso a paso la fusión de regiones.

El programa principal que ejecuta la interfaz gráfica recibe el nombre de *Postprocess.m*. En el Anexo A, podemos ver las llamadas que realiza esta función a las distintas funciones que ejecutan los algoritmos descritos anteriormente.

CAPÍTULO 5

EXPERIMENTOS Y EVALUACIÓN

En este capítulo evaluaremos los algoritmos desarrollados para el post-procesado de imágenes junto con diferentes técnicas de segmentación de forma subjetiva y objetiva.

5.1 EXPERIMENTOS

5.1.1 Introducción

En este apartado del capítulo vamos a realizar dos tipos de pruebas.

En primer lugar, evaluaremos de forma subjetiva cómo, en algunos casos, con un sólo parámetro se obtienen mejores resultados que con el uso de todos ellos simultáneamente, en función del tipo de objeto que contiene la imagen que se va a post-procesar. Para ello, sólo utilizaremos imágenes con un sólo objeto y una única técnica de segmentación que será el algoritmo *kmeans* con 4 clases.

Más adelante, demostraremos cómo mejora esta herramienta la segmentación. Para ello, nos basaremos en las pruebas realizadas para el software de segmentación incluido en la herramienta, tomaremos los resultados fracasados en [25] y veremos si el algoritmo de segmentación es realmente ineficiente o si, por el contrario, la segmentación necesita del bloque de post-procesado.

Para llevar a cabo la segunda parte de estos experimentos, se han seleccionado varias imágenes de la base de datos de la Universidad de Berkeley (*Ref., [26]*). Esta base de datos se utilizará también en las pruebas objetivas realizadas en el siguiente apartado del capítulo.

Con estos experimentos tenemos como objetivo evaluar las prestaciones que nos ofrece la herramienta de post-procesado realizada. No hay que olvidar que se trata de unas pruebas subjetivas que consisten en comparar cualitativamente, no cuantitativamente, los resultados obtenidos de la segmentación con los resultados que se obtienen al añadir un bloque de post-procesado, así como el post-procesado en función de los parámetros seleccionados.

Siguiendo con el tipo de pruebas descrito en [25], utilizaremos la misma clasificación para la valoración de los resultados siendo la escala utilizada la siguiente:

- Muy bueno
- Bueno
- Aceptable
- Malo
- Muy Malo

Estas valoraciones se toman con referencia a la segmentación que realizaría una persona. La parte más alta de la escala, “muy bueno”, se corresponde con los resultados que más se asemejen con la segmentación realizada por una persona. La valoración “bueno” pertenece a los resultados que, aún teniendo algunos fallos, siguen proporcionando un alto grado de similitud con la segmentación que realizaría una persona. Utilizaremos “aceptable” siempre que se pueda discernir en una imagen el objeto del fondo aunque existan bastantes errores. Por último, las valoraciones “malo” y “muy malo” se corresponden con los resultado que no aporten ningún significado y, por lo tanto, no sea posible discernir diferentes objetos ni diferenciarlos del fondo. (*Ref. [25]*)

5.1.2 Selección de parámetros

Para este tipo de pruebas, como se mencionó en la introducción del capítulo, el algoritmo utilizado es el *kmeans* con 4 clases sobre las bandas R, G y B.

En este experimento vamos a ver los valores de los descriptores antes de realizar el post-procesado y una vez que este se ha realizado.

Vamos a dividir la sección en diferentes casos, en cada caso de estudio se evaluará una imagen en concreto.

5.1.2.1 Caso 1

Para el caso de estudio número 1 vamos a utilizar la imagen de referencia de la figura 5.1. Esta imagen contiene un erizo.

Figura 5.1. Imagen de referencia para el caso 1.

En primer lugar, realizamos la segmentación de la imagen (figura 5.2) para poder estudiar los valores de cada parámetro.

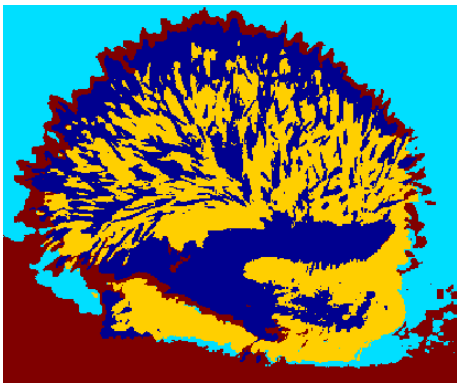


Figura 5.2. Imagen segmentada en cuatro regiones para el caso 1.

Ahora, vamos a ver el resultado de cada descriptor para cada una de las regiones que contiene la imagen.

Tabla 5.1. Resultado de los parámetros para el caso de estudio 1.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,99421195	0,53602286	0,99186992	0,35305263	0,0983844
Región 2 (turquesa)	0,96590173	0,71064025	0,97435897	0,26761484	0,41494134
Región 3 (amarillo)	0,99323434	0,50945654	0,98947368	0,38107328	0,21694866
Región 4 (rojo)	0,98745688	0,76878884	0,98780488	0,1928804	0,27386242

Vemos que, en general, los resultados obtenidos se aproximan a 1 mientras que, lo deseado es que se aproximen a 0.

A continuación, haremos el post-procesado, de la imagen anterior, con todos los parámetros posibles. Esto es, el número de componentes conectadas, la compacidad, la excentricidad, la rectangularidad y convexidad. La imagen que obtenemos se muestra en la figura siguiente.

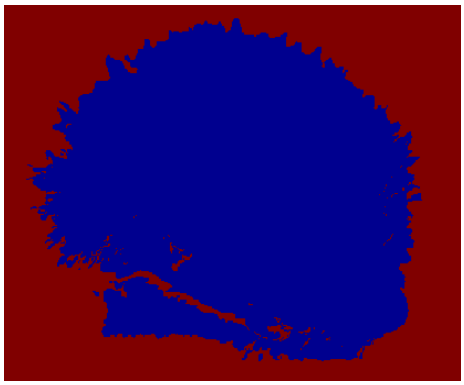


Figura 5.3. Imagen post-procesada usando todos parámetros para el caso 1.

Una vez post-procesada la imagen, el valor de cada parámetro para cada una de las dos regiones se muestra en la tabla 5.2.

Tabla 5.2. Resultado de los parámetros después del post-procesado para el caso de estudio 1.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,84157351	0,11750607	0,9	0,69767442	0,00057856
Región 2 (rojo)	0,78832907	0,55106561	0,98507463	0,44893439	0,00073809

Excepto en el caso de la rectangularidad, todos los descriptores han disminuido notablemente su valor.

La excentricidad ha sido el parámetro que más ha disminuido, alcanzando casi el valor ideal. Esto se debe a que la figura final obtenida se asemeja a un círculo y recordemos que la excentricidad normalizada para este tipo de figuras es cero.

Cabe destacar que, debido a la segmentación, siguen existiendo píxeles de una región dispersados. Por ello, el número de componentes o la compacidad no se aproximan a cero.

5.1.2.2 Caso 2

En este caso, vamos a estudiar una imagen con un objeto sobre-segmentado. La imagen de referencia se muestra en la figura 5.4.

Figura 5.4. Imagen de referencia para el caso 2.

El resultado de la segmentación de esta imagen (figura 5.5) es dos regiones muy dispersas que se corresponden al objeto (pirámide) y dos regiones más uniformes que se corresponden con el fondo (cielo).

Figura 5.5. Imagen segmentada en cuatro regiones para el caso 2.

Los valores de cada región de la imagen anterior los podemos ver en la tabla siguiente.

Tabla 5.3. Resultado de los parámetros para el caso de estudio 2.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,8513124	0,71518905	0,97959184	0,21776497	0,68956018
Región 2 (turquesa)	0,86592334	0,6058627	0,98275862	0,35068905	0,79734547
Región 3 (amarillo)	0,99838004	0,50256854	0,99901478	0,30348032	0,46825158
Región 4 (rojo)	0,99814021	0,5112179	0,99887387	0,29931669	0,47393436

Los valores anteriores reflejan el caso que estudiamos, que se corresponden con regiones poco compactas. Si realizamos el post-procesado de la imagen de la figura 5.5, el resultado que obtenemos es el siguiente,

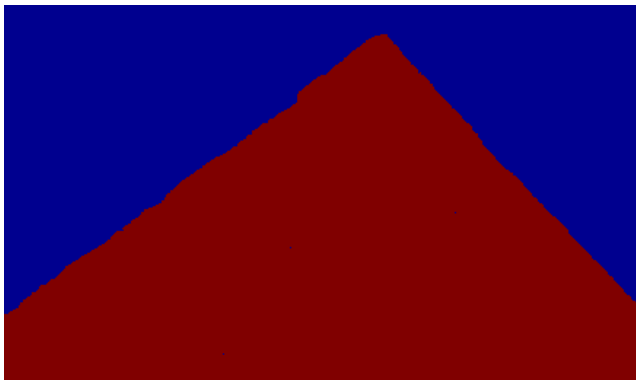


Figura 5.6. Imagen post-procesada usando todos parámetros para el caso 2.

Una vez realizado el post-procesamiento, los valores de los descriptores para cada región son,

Tabla 5.4. Resultado de los parámetros después del post-procesado para el caso de estudio 2.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,71008884	0,48561807	0,75	0,27900701	0,00449093
Región 2 (rojo)	0,41263888	0,01575377	0	0,30279701	0,00191191

En este caso, el valor de todos los parámetros se ha reducido. En el caso de las componentes conectadas, hemos alcanzado el valor óptimo para una de las

regiones. La compacidad también demuestra que las regiones son ahora más compactas.

5.1.2.3 Caso 3

En los casos anteriores, el resultado del post-procesamiento era el deseado. En ocasiones, esto no se consigue, pero sí podemos alcanzarlo si estudiamos el tipo de región que queremos unificar.

En la siguiente figura se muestra la imagen de referencia para el caso de estudio número 3.

Figura 5.7. Imagen de referencia para el caso 3.

Segmentamos la imagen en cuatro regiones (figura 5.8). A continuación, vemos los valores que alcanza cada descriptor para cada una de las cuatro regiones (tabla 5.5).

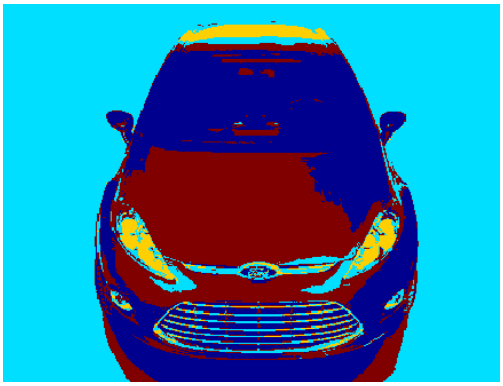


Figura 5.8. Imagen segmentada en cuatro regiones para el caso 3.

Tabla 5.5. Resultado de los parámetros para el caso de estudio 3.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,98753246	0,51350394	0,99019608	0,42905405	0,23170538
Región 2 (turquesa)	0,97857865	0,47440554	0,99528302	0,22559446	0,35380588
Región 3 (amarillo)	0,98963542	0,94303412	0,99342105	0,04717557	0,24330576
Región 4 (rojo)	0,99354471	0,62422973	0,99756098	0,32614125	0,12060084

Realizamos el post-procesamiento con todos los parámetros anteriores y el resultado obtenido (figura 5.9) no es muy satisfactorio. El resultado esperado sería la fusión de las regiones 1, 3 y 4. Pero, en lugar de eso, se han fusionado [3 2] y [1 2].

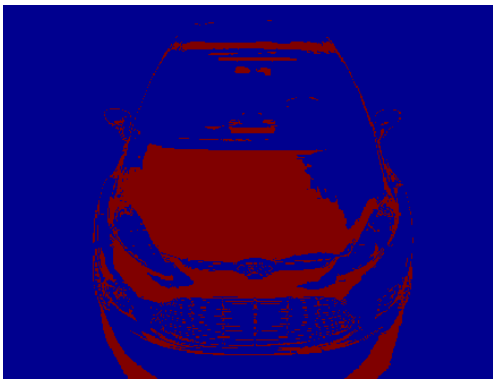


Figura 5.9. Imagen post-procesada usando todos parámetros para el caso 3.

Los valores obtenidos después de la fusión de regiones se muestran en la tabla 5.6.

Tabla 5.6. Resultado de los parámetros después del post-procesado para el caso de estudio 3.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,66748326	0,20310261	0,97297297	0,19689739	0,00085483
Región 2 (rojo)	0,99354471	0,62422973	0,99756098	0,32614125	0,00034371

Los resultados anteriores no son muy buenos. Se han mejorado para la región uno pero, no para la región dos. Esta decisión se ha realizado debido al

parámetro de la compacidad y al número de componentes conectadas, ya que si vemos el resultado de estos descriptores en la tabla 5.5 eran los mayores.

Ahora, vamos a utilizar un único parámetro para realizar el post-procesamiento. Este parámetro va a ser la envoltura convexa ya que la forma que se pretende obtener en la imagen segmentada (figura 5.8) es una forma ovalada. El resultado obtenido es el siguiente,

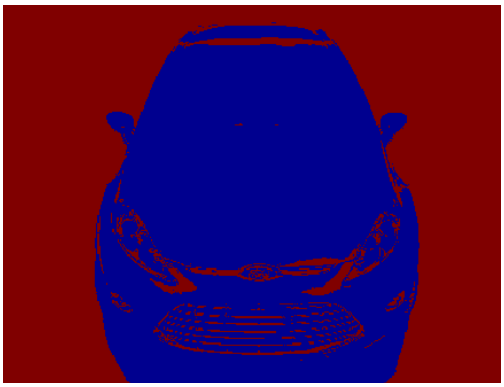


Figura 5.10. Imagen post-procesada usando el parámetro de convexidad para el caso 3.

Como vemos en la figura anterior, el resultado es el óptimo. No se había alcanzado antes ya que el parámetro utilizado en este caso, no era uno de los que alcanzaban un valor mayor.

A continuación, vamos a calcular los valores de cada parámetro para cada una de las regiones de la imagen de la figura 5.10.

Tabla 5.7. Resultado de los parámetros después del post-procesado con la convexidad para el caso de estudio 3.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,70402298	0,12399441	0,92857143	0,16179828	0,14215776
Región 2 (rojo)	0,97857865	0,47440554	0,99528302	0,22559446	0,35380588

En media, todos los parámetros han reducido su valor respecto al anterior post-procesamiento. Esto se produce, como hemos explicado anteriormente, porque la envoltura convexa, en un principio, no era uno de los parámetros de

mayor valor pero, debido a la forma del objeto de este parámetro sí es uno de los descriptores más significativos para extraer una región con estas características.

5.1.2.4 Caso 4

El siguiente caso es parecido al anterior, en el que el uso de todos los parámetros simultáneamente no funciona correctamente mientras que, un solo parámetro es capaz de diferenciar de forma óptima el objeto.

La imagen de referencia para este caso de estudio se muestra en la figura 5.11.

Figura 5.11. Imagen de referencia para el caso 4.

A continuación, realizamos la segmentación (figura 5.12) y mostramos los valores de las características de cada región (tabla 5.8).

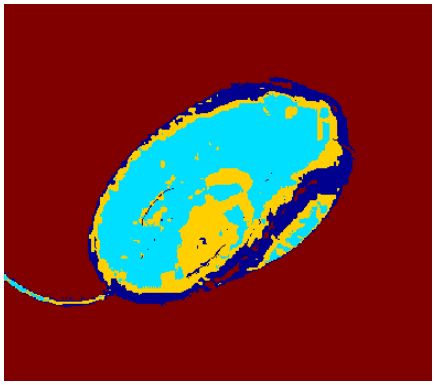


Figura 5.12. Imagen segmentada en cuatro regiones para el caso 4.

Tabla 5.8. Resultado de los parámetros para el caso de estudio 4.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,96804367	0,83353093	0,98333333	0,10060259	0,42042709
Región 2 (turquesa)	0,91195295	0,47424623	0,95833333	0,31652356	0,53352296
Región 3 (amarillo)	0,9682592	0,73828099	0,98113208	0,15920983	0,43133719
Región 4 (rojo)	0,84294738	0,25561224	0,97435897	0,74438776	0,11927624

Si realizamos el post-procesado usando todos los descriptores, el resultado es el siguiente,

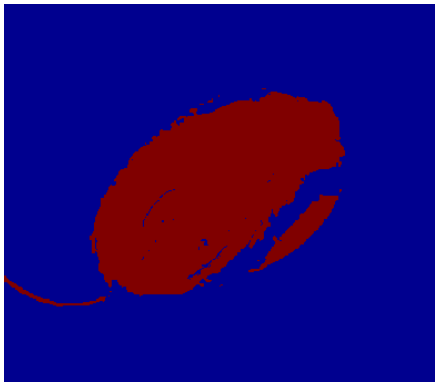


Figura 5.13. Imagen post-procesada usando todos parámetros para el caso

Los valores obtenidos después de la fusión de regiones se muestran en la tabla 5.9.

Tabla 5.9. Resultado de los parámetros después del post-procesado para el caso de estudio 4.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,63384414	0,20481447	0,96428571	0,79518553	0,11678528
Región 2 (rojo)	0,81568024	0,26545346	0,91666667	0,4477954	0,45879235

La imagen resultado de la figura 5.13 no es el fusionado esperado ya que la región 1 se mezcla con el fondo y no con el objeto.

Sin embargo, si observamos el objeto de la imagen, este ha sido sobre-segmentado y el número de componentes conectadas es un descriptor significativo.

Por lo que vamos a proceder a realizar el post-procesado solamente con el número de componentes conectadas.

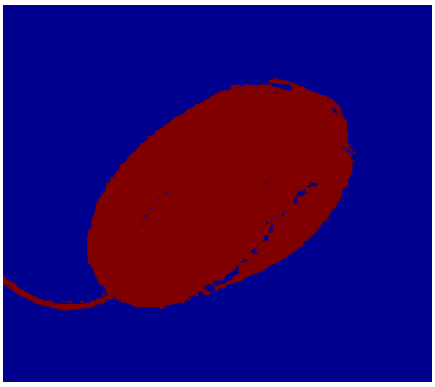


Figura 5.14. Imagen post-procesada usando el parámetro del número de componentes conectadas para el caso 4.

Como vemos en la figura anterior, obtenemos el resultado esperado. A continuación, vamos a calcular los valores de cada parámetro para cada una de las regiones de la imagen de la figura 5.14.

Tabla 5.10. Resultado de los parámetros después del post-procesado con el número de componentes conectadas para el caso de estudio 4.

	Compacidad	Envoltura convexa	Componentes conectadas	Rectangularidad	Excentricidad
Región 1 (azul)	0,84294738	0,25561224	0,97435897	0,74438776	0,11927624
Región 2 (rojo)	0,6682227	0,1623347	0,875	0,40622795	0,44002216

Si comparamos los resultados de la tabla 5.10 con los resultados de la tabla 5.9, observamos que, en el caso de la compacidad, antes el fondo era más compacto que el objeto mientras que, ahora es al contrario. El número de componentes conectadas del objeto también ha disminuido en el segundo caso. Y, en general, las propiedades del objeto se han mejorado.

En el primer post-procesado de este caso, no se alcanzaba el resultado óptimo mientras que se mejoraban los resultados de ambas regiones finales (región objeto y región fondo). En el segundo, las propiedades del objeto han sido mejoradas utilizando el número de componentes conectadas.

5.1.2.5 Discusión

En los casos anteriores, hemos estudiado cómo con algunos parámetros en particular se obtienen mejores resultados según el tipo de objeto de la imagen. Por ello, en la interfaz gráfica se pueden seleccionar estos parámetros de forma independiente.

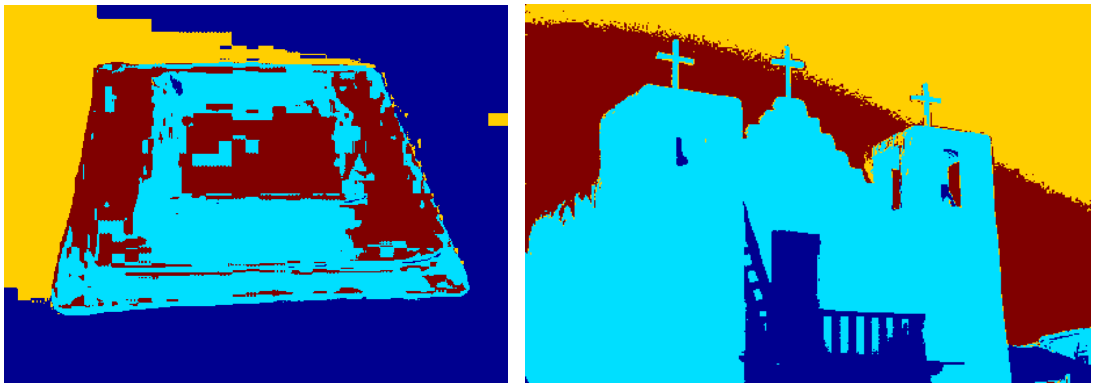
Los experimentos realizados hasta ahora quieren ofrecer un grado de comprensión mayor para el usuario que haga uso de la herramienta de post-procesado. Pero, si lo que queremos es una herramienta automática el conjunto de todos los parámetros, en la mayoría de los casos, logran un resultado satisfactorio.

A continuación mostramos dos ejemplos. Las imágenes de referencia las vemos en la siguiente figura.

(a) (b)

Figura 5.15. Imágenes de referencia para la discusión.

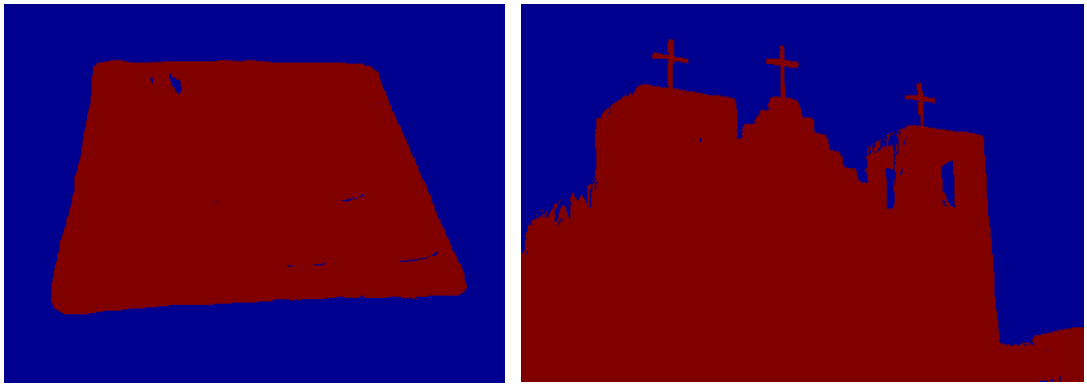
Realizamos, como hasta ahora, la segmentación de las imágenes de la figura 5.15 en cuatro regiones y los resultados obtenidos son los siguientes,



(a) (b)

Figura 5.16. Segmentación en cuatro regiones de las imágenes de referencia para la discusión.

Finalmente, realizamos el post-procesado con todos los parámetros que nos permite la herramienta desarrollada en este proyecto. El resultado de este post-procesado sobre la segmentación en cuatro regiones de la figura 5.16, se muestra en la figura 5.17.



(a) (b)

Figura 5.17. Post-procesado de la segmentación en cuatro regiones de las imágenes de referencia para la discusión, usando todos los parámetros.

Observamos que el resultado obtenido es el óptimo en base a la segmentación realizada.

Por lo que, si utilizamos la herramienta de forma automática no es necesario ir procesando imagen a imagen estudiando las características de las regiones que se obtienen de la segmentación, podemos conseguir buenos resultados haciendo uso simultáneo de todos los parámetros que se ofrecen.

5.1.3 Mejora de los algoritmos de segmentación

En esta sección vamos a comprobar cómo, en la mayoría de los casos, el bloque de post-procesado mejora la segmentación realizada por los algoritmos de la librería desarrollada en [25].

La segmentación realizada por cada algoritmo depende en gran medida de la selección de características de la imagen que hagamos previamente. Así pues, vamos a dividir la sección en dos, una parte se realizará con el mismo algoritmo de segmentación y diferentes características y en otra, evaluaremos la mejora respecto a los diferentes algoritmos de segmentación usando las mismas características.

Para los experimentos, y puesto que la herramienta de segmentación tiene cantidad de opciones de extracción de características, vamos a estudiar las características que ofrecían peores resultados en las pruebas realizadas en [25]. Como se mencionó en la introducción de este capítulo vamos a usar la misma escala de evaluación.

La imagen de referencia utilizada se muestra en la figura 5.18. Esta imagen se ha tomado de la base de datos de la Universidad de Berkeley (*Ref. [26]*).

Figura 5.18. Imagen de referencia para evaluar los algoritmos de la herramienta de post-procesado.

A continuación, pasamos a estudiar cada caso.

5.1.3.1 Selección de características

Para esta sección, vamos a ir utilizando las características que ofrecen peores resultados. El algoritmo de segmentación utilizado es el algoritmo basado en la técnica de agrupamiento *EM* (Maximization of expectation)[11] por su rapidez de cómputo y debido a que de esta forma se realizaron las pruebas que vamos a continuar. La elección del algoritmo de segmentación, en este caso, no es demasiado importante ya que vamos a utilizar el mismo algoritmo para todo el conjunto de pruebas.

Características de color:

En primer lugar vamos a estudiar el espacio de color **RGB**. Como vemos en [25], la segmentación en dos regiones mediante el algoritmo EM de las bandas R, G y B nos da un resultado "malo" (figura 5.19) ya que no se puede distinguir el objeto, en este caso las flores, del fondo debido a que las bandas RGB no aportan información suficiente para diferenciarlo.

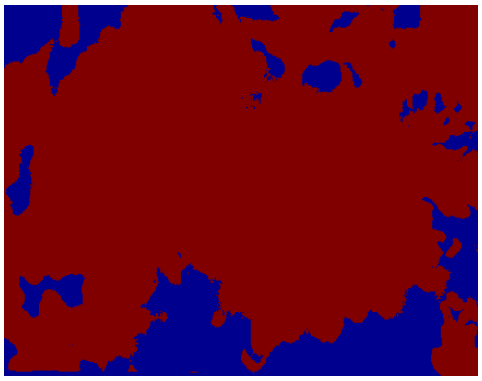


Figura 5.19. Segmentación de las componentes RGB en dos regiones.

Una posible solución que se ofrece en [25] es aumentar el número de clases a tres. En tal caso, obtenemos la imagen de la figura 5.20.

[11] Técnica de agrupamiento EM, ver capítulo 9 de [25].

Figura 5.20. Segmentación de las componentes RGB en tres regiones.

De esta forma, si se diferencian las flores pero, el fondo ha sido sobre-segmentado en dos. A continuación, realizamos el post-procesamiento de la imagen y obtenemos el resultado de la figura 5.21.

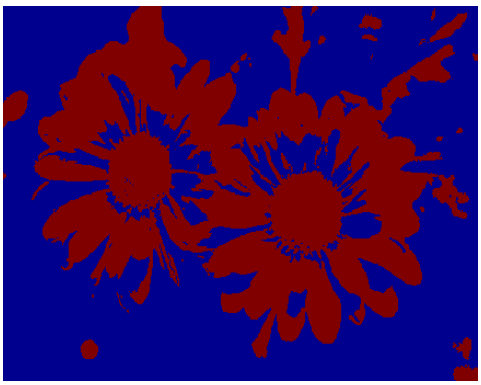


Figura 5.21. Post-procesado de la segmentación de las componentes RGB en tres regiones.

Con el post-procesado, el resultado ha pasado de ser "muy malo" (figura 5.19) a "bueno" (figura 5.21), vemos que parte del fondo se ha mezclado con el objeto aunque, como se puede observar, esto es culpa de la segmentación y no del post-procesado.

Otro espacio de color que necesita de la herramienta de post-procesado es el espacio **YCbCr**. El resultado que nos da la segmentación en dos regiones (figura 5.22) con estas características es clasificado como "muy malo".

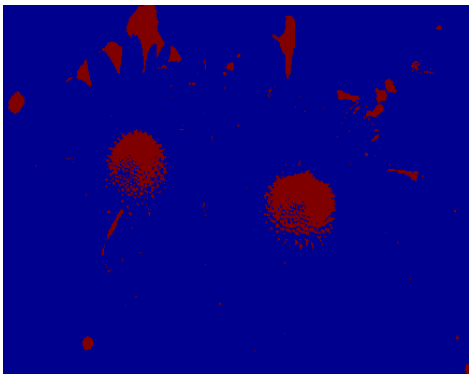


Figura 5.22. Segmentación de las componentes YCbCr en dos regiones.

Si se aumenta el número de clases de la segmentación a cuatro regiones (figura 5.23), el resultado del post-procesado (figura 5.24) es “aceptable” ya que todavía se sigue mezclando el fondo con el objeto.

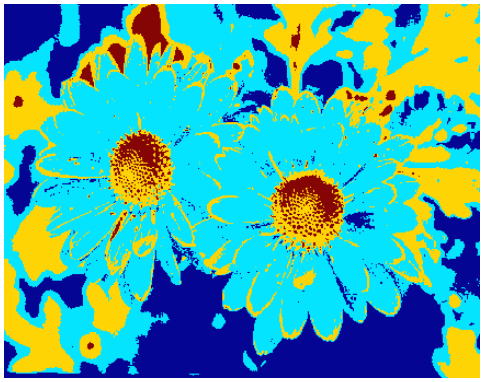


Figura 5.23. Segmentación de las componentes YCbCr en cuatro regiones.

Figura 5.24. Post-procesado de la segmentación de las componentes YCbCr en cuatro regiones.

Sin embargo, con el espacio de color **XYZ** el resultado empeora si aumentamos el número de clases. En la figura 5.25 se puede observar como la segmentación de las bandas X, Y y Z en dos regiones es "bueno".

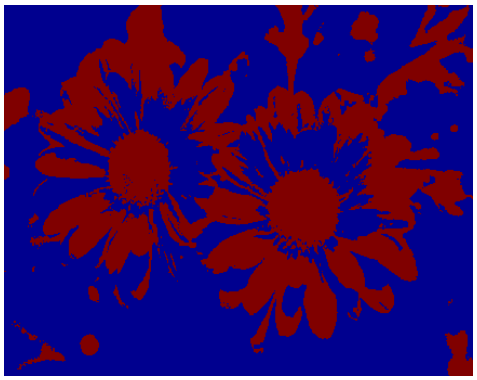


Figura 5.25. Segmentación de las componentes XYZ en dos regiones.

Mientras que, si aumentamos el número de clases a tres (figura 5.26), y realizamos el post-procesado el resultado es "aceptable" (figura 5.27).

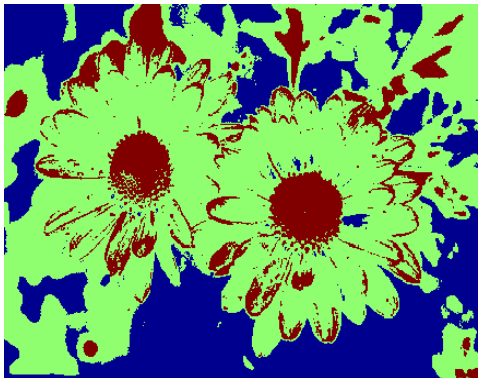


Figura 5.26. Segmentación de las componentes XYZ en tres regiones.

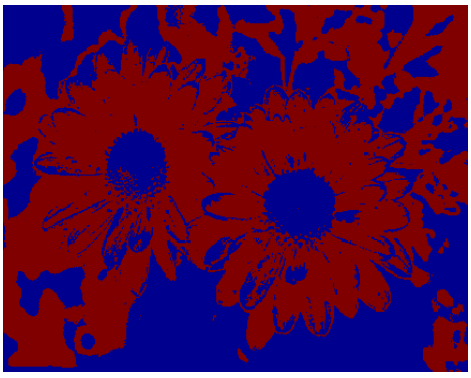


Figura 5.27. Post-procesado de la segmentación de las componentes XYZ en tres regiones.

Características de textura:

En algunas ocasiones, las componentes de color no son suficientes para realizar la extracción de características. Como vemos en la imagen de la figura 5.28, extraída también de la base de datos de la Universidad de Berkeley (*Ref. [26]*), tanto el objeto (leopardo) como el fondo comparten características de color, mientras que, tienen texturas diferentes.

Figura 5.28. Imagen de referencia para evaluar las características de textura.

El resultado de la segmentación depende de la característica de color elegida para realizar los cálculos de textura. Como venimos haciendo hasta ahora, vamos a utilizar la componente de luminancia, Y, ya que con esta banda de color se han realizado las pruebas de textura en [25].

La herramienta de segmentación nos permite extraer varías características de textura. Pero, por simplificar, en este proyecto sólo vamos a usar el rango.

En primer lugar, realizamos la segmentación del rango de la componente de luminancia en dos regiones (figura 5.29) y observamos que, el resultado se clasifica como “malo”.

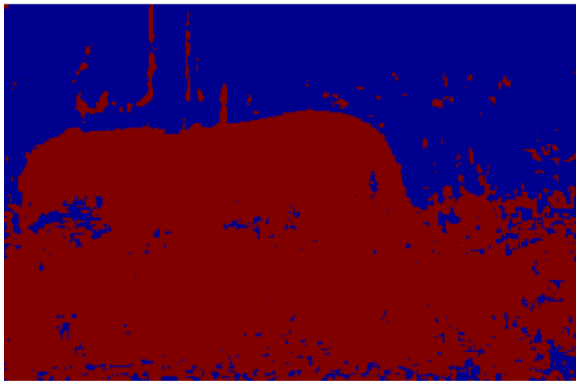


Figura 5.29. Segmentación del rango de la componente de luminancia en dos regiones.

Si aumentamos el número de clases a tres, el resultado mejora y se clasifica como “aceptable”. El resultado de esta segmentación se refleja en la figura siguiente.

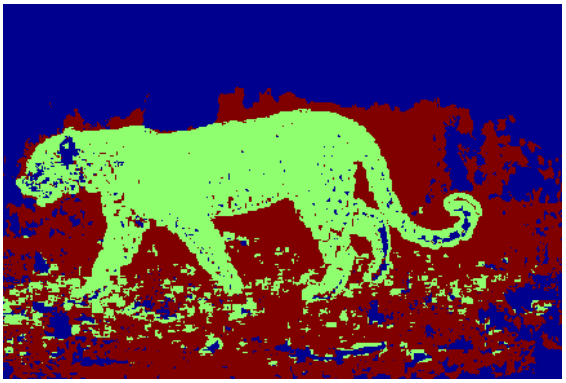


Figura 5.30. Segmentación del rango de la componente de luminancia en tres regiones.

Ahora, vamos a post-procesar la imagen de la figura anterior y el resultado alcanzado es el siguiente,

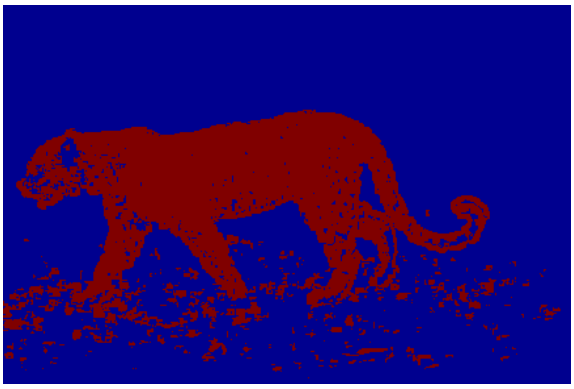


Figura 5.31. Post-procesado de la segmentación del rango de la componente de luminancia en tres regiones.

De esta forma, el resultado del post-procesado lo clasificamos como "bueno". Una vez más, comprobamos que nuestra herramienta mejora la segmentación.

5.1.3.2 Algoritmo kmeans y algoritmo EM

En esta sección vamos a evaluar los dos algoritmos de segmentación por agrupamiento que nos ofrece la herramienta de segmentación integrada en nuestro software. Estos algoritmos son el algoritmo *kmeans* y el algoritmo *EM*. Vamos a estudiarlos extrayendo de la imagen las características de color del espacio HSV.

La imagen de referencia utilizada vuelve a ser la imagen mostrada en la figura 5.18 (flores).

Algoritmo *kmeans*:

Para el algoritmo *kmeans*, si realizamos la segmentación en dos regiones obtenemos un resultado "malo", figura 5.32. Sólo distingue los pétalos de las flores como objeto.

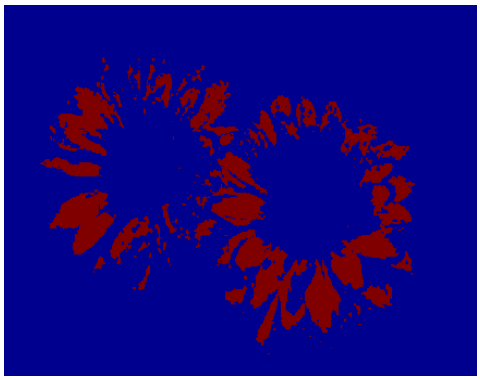


Figura 5.32. Segmentación con algoritmo *kmeans* en dos regiones.

Sin embargo, si aumentamos el número de clases a cuatro (figura 5.33) y, después, realizamos el post-procesado (figura 5.34) el resultado pasa a ser "muy bueno" ya que extrae como objeto lo que una persona distinguiría como tal en esta imagen.

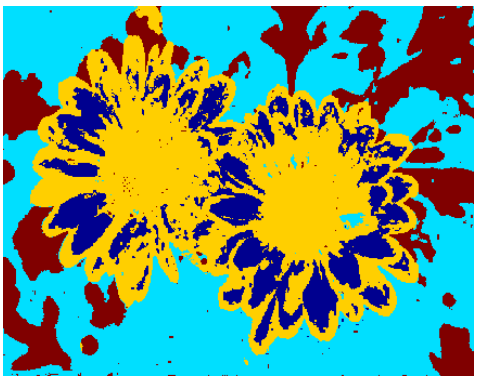


Figura 5.33. Segmentación con algoritmo *kmeans* en cuatro regiones.

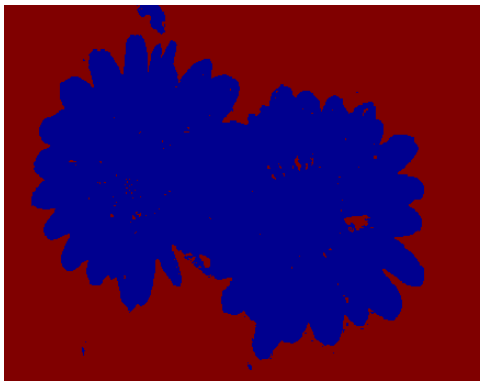


Figura 5.34. Post-procesado de la segmentación del algoritmo *kmeans* en cuatro regiones.

Algoritmo *EM*:

A continuación, vamos a realizar el mismo estudio que con el algoritmo anterior. En este caso, el algoritmo *EM* ofrece una segmentación, en dos regiones, "mala" y exactamente igual al caso del algoritmo kmeans, lo observamos en la siguiente figura,

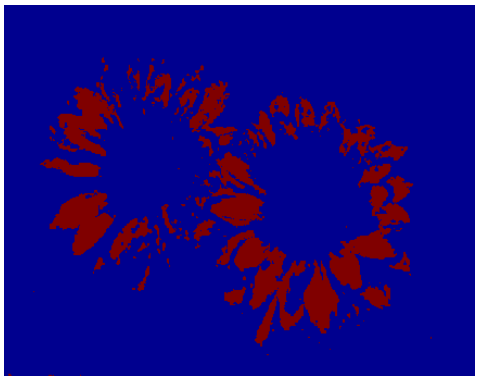


Figura 5.35. Segmentación con algoritmo *EM* en dos regiones.

Como vemos, sólo se puede distinguir parte del objeto. Sin embargo, si aumentamos el número de clases a cuatro, la segmentación se queda prácticamente igual, mientras que si aumentamos el número de a clases a 6 (figura 5.36), la segmentación mejora y se puede evaluar como "aceptable".

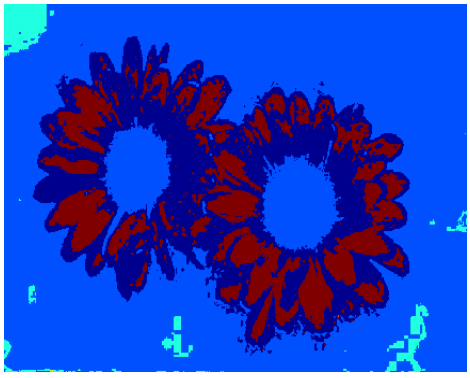


Figura 5.36. Segmentación con algoritmo *EM* en seis regiones.

Si, posteriormente, aplicamos el post-procesado a la imagen anterior, el resultado se convierte en "malo", siendo idéntico al caso de la segmentación en dos regiones. Observamos este resultado en la siguiente figura,

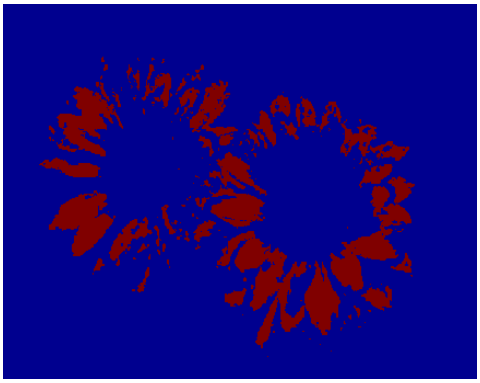


Figura 5.37. Post-procesado de la segmentación del algoritmo *EM* en cuatro regiones.

Por lo tanto, por mucho que aumentemos el número de clase el algorimo *kmeans* nos ofrece mejores resultados que el algoritmo *EM*. Por esta razón, el algoritmo utilizado para las pruebas objetivas será el algoritmo *kmeans*.

5.1.4 Conclusiones de los experimentos

Con los experimentos desarrollados en las dos secciones anteriores se ha pretendido evaluar cualitativamente el proceso de post-procesado de imagen. Hemos demostrado cómo, en la mayoría de los casos, la segmentación se mejora si aumentamos el número de clases y aplicamos posteriormente el bloque de post-procesado.

En el primer apartado, hemos realizado un estudio de los parámetros de la herramienta de post-procesado. En algunos casos, es más eficiente utilizar un parámetro de forma independiente que el conjunto de todos ellos. Esto se debe, en primer lugar, a la naturaleza del objeto que contiene la imagen y, en segundo lugar, a la segmentación que se realiza.

Así pues, si queremos obtener el mejor resultado de una imagen en concreto, nuestra herramienta ofrece libertad, a través de la interfaz gráfica de usuario, para post-procesar la imagen con un/os parámetro/s específico/s.

Pero, en la segunda sección de este apartado, en todas las pruebas que se han realizado se ha hecho uso de todos los parámetros por lo que, de forma general, el uso de todos ellos ofrece buenos resultados. Hemos extraído diferentes

características de la imagen original con ayuda de la herramienta de segmentación integrada en este proyecto, esto hacía que, para el mismo método de segmentación, el resultado fuera diferente. Sin embargo, los resultados obtenidos después del post-procesado han sido mayoritariamente satisfactorios.

5.2 PRUEBAS AUTOMÁTICAS

5.2.1 Introducción

En este apartado vamos a hablar de las pruebas que se han realizado para evaluar nuestra herramienta de forma cuantitativa. Para ello, hemos utilizado un software desarrollado por la Universidad de Berkeley, *The Berkeley Segmentation Dataset and Benchmark* y, una base de datos creada por la misma universidad (*Ref. [26]*).

El principal objetivo de este software es proporcionar un punto de referencia y evaluación (*benchmark*) para la investigación sobre la segmentación de imágenes y detección de bordes. Con este propósito, la herramienta nos proporciona 12.000 segmentaciones realizados a mano por 30 personas diferentes. La mitad de estas segmentaciones se obtuvieron presentando a estas personas imágenes a color y, la otra mitad, a escala de grises. Las imágenes de la base de datos se dividen en un conjunto de entrenamiento de 200 imágenes y otro de test de 100 imágenes.

La colección de bordes marcados por diversas personas sobre las imágenes es lo que constituye el marco de bordes válidos. Posteriormente, se toma la salida de un cierto algoritmo para una imagen, asumiendo que esa salida es un mapa de clasificación de bordes suave de un píxel de anchura, que toma valores de cero a uno donde los valores mayores aportan mayor probabilidad de la existencia de un borde. La tarea de este software es determinar la calidad de ese mapa de clasificación de bordes en comparación con el marco de referencia válido (*Ref. [26]*).

Tradicionalmente, se podría binarizar el mapa de clasificación eligiendo un umbral determinado para obtener una imagen con ceros para el fondo y unos para los bordes de los objetos. Sin embargo, hay dos problemas relativos a la umbralización de un mapa de clasificación. Por un lado, el umbral óptimo depende de la aplicación, y el objetivo de este software es proporcionar un estándar de comparación, *benchmark,* que sea útil para diferentes aplicaciones y, por otro lado, el proceso de umbralización sobre una característica de bajo nivel como pueden ser los bordes de una imagen, es subóptimo debido a que se pierde mucha

información. Por estas razones, este software no trabaja sobre mapas de clasificación de bordes no umbralizados.

No obstante, es necesario umbralizar el mapa de clasificación de bordes para compararlo con los bordes detectados por personas, pero esto se realiza en varios niveles. En cada nivel, se calculan dos cantidades, *precision* y *recall* (precisión y exhaustividad), y se traza una curva para el algoritmo, curva de *precisión-recall*.

La *precision* es la probabilidad de que un pixel de un borde generado por una máquina sea un píxel verdadero, es decir, que pertenezca al borde y *recall* es la probabilidad de que un píxel verdadero sea detectado.

En la práctica, estas cantidades nos aportan información de cuánto ruido hay en la salida del detector (*precision*) y cuántos bordes verdaderos son detectados (*recall*). A medida que el umbral del detector cambia, varía la compensación entre estas cantidades, compensación entre las pérdidas y los falsos positivos. Para tener una medida única de la calidad del algoritmo se utiliza la *medida-F* (*F-measure*), que es la distancia harmónica entre *precision* y *recall* y se calcula mediante la siguiente ecuación,

$$F = 2 \times \frac{precision \times recall}{precision + recall} \qquad (5.1)$$

El resultado de las pruebas depende del umbral utilizado para binarizar el mapa de bordes. Este umbral es elegido automáticamente por la herramienta de las pruebas. Este proceso es lento porque se prueba con diversos valores de umbral con los que dibuja la curva *precisión-recall* para cada mapa de bordes de cada algoritmo, y finalmente se queda con aquel que ha obtenido un valor menor de la *medida-F*. La herramienta de evaluación también ofrece la posibilidad de hacer las pruebas en modo rápido, fijando el valor del umbral a 0.5 para todas las imágenes, de modo que ya no se puede dibujar la curva *precisión-recall* porque obtenemos un solo punto en la gráfica.

La curva *precisión-recall* cualitativamente es similar a la curva ROC, muestra la misma relación entre pérdidas y falsos positivos pero, con diferentes ejes. Los ejes de la curva ROC representan la tasa de falsa alarma, que es la probabilidad de

que un verdadero negativo sea etiquetado como un falso positivo y, el *recall*. Sin embargo, la curva ROC no es apropiada para cuantificar detección de bordes ya que depende de la resolución de la imagen. La *precisión* no tiene este problema dado que se normaliza por el número de positivos y no por el número de verdaderos negativos (*Ref. [26]*).

5.2.2 Desarrollo de las pruebas

Para la realización de las pruebas objetivas ha habido que adaptar la salida de los algoritmos y seguir una serie de pasos, todos ellos indicados en [26].

En primer lugar, hay que procesar las imágenes de la base de datos para después ejecutar la herramienta de la Universidad de Berkeley. Es importante señalar que esta evaluación se ha realizado con el conjunto de test de la base de datos.

Para las imágenes de color se ha decidido usar las componentes a y b del espacio de color L*a*b y las componentes RGB y para las imágenes en escala de grises la componente de luminancia, sobre estas bandas se ha aplicado el algoritmo de segmentación por agrupamiento *kmeans* en tres clases y, posteriormente, se ha realizado el post-procesamiento haciendo uso de todos los parámetros.

Con respecto a las imágenes en escala de grises, al tener únicamente una banda de características, no tenemos opciones de combinación, por lo que se ha utilizado únicamente la componente de luminancia. Sin embargo, para las imágenes a color se han decidido utilizar los espacios L*a*b y RGB ya que los mejores resultados que se obtuvieron en [25] fueron haciendo uso de estos espacios de color.

Lo mismo sucede con el uso del algoritmo *kmeans*, además que, como vimos el apartado anterior, ofrece mejores resultados que haciendo uso del algoritmo de segmentación basado en la técnica de agrupamiento *EM*.

Para el post-procesado, se han utilizado todos los parámetros ya que la base de datos de la Universidad de Berkeley contiene imágenes muy variadas. En la

siguiente figura podemos ver algunas de las imágenes del conjunto de test de la base de datos.

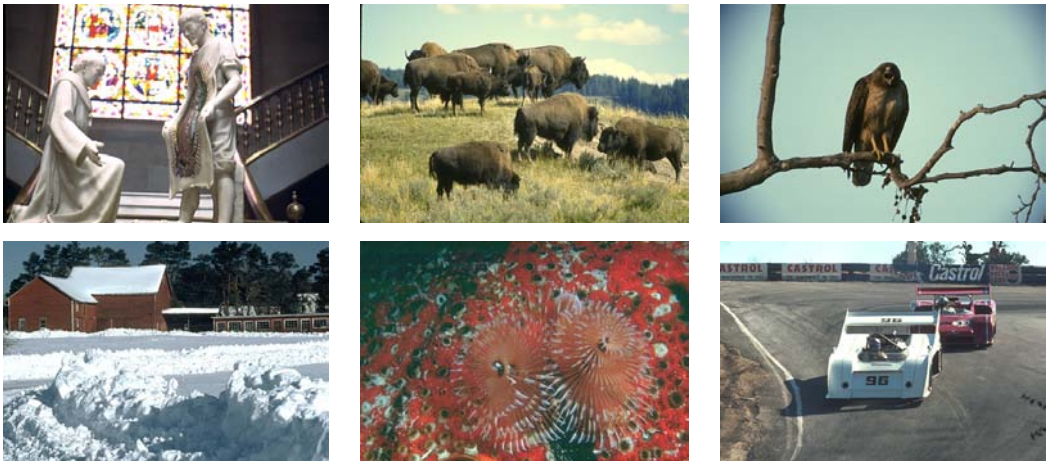


Figura 5.38. Imágenes del conjunto de test de la base de datos.

Una vez conseguida la imagen post-procesada, se procede a utilizar los diferentes algoritmos de detección de bordes implementados por la herramienta de segmentación integrada en este proyecto. Pero, puesto que estos algoritmos nos aportan una detección de bordes con un resultado ya umbralizado, lo que no cumple con las especificaciones del software de evaluación, se ha añadido a la salida una etapa de filtrado para que se adaptaran los resultados a los requeridos. Este filtrado consiste en un filtro paso-bajo de los bordes con una ventana deslizante, de forma que se consigue obtener una detección de bordes blanda.

Esto significa que a la salida tendremos una imagen con ceros en el fondo y con valores comprendidos entre [0, 1] en los bordes, que indican la probabilidad de que ese borde pertenezca realmente a un borde verdadero, es decir, un borde que sería marcado por una persona.

Un aspecto importante es la elección del tamaño de la ventana para el filtrado. En este caso, se ha utilizado una ventana de 3x3 ya que nuestro objetivo es obtener un aspecto de promediado del borde en una vecindad próxima.

Una vez realizado todos estos pasos, para poder comparar las imágenes con las segmentaciones realizadas por personas hay que guardar los resultados en formato BMP. Asimismo, estos resultados deben tener las mismas dimensiones que las imágenes que nos proporciona la herramienta de evaluación (481x321 píxeles).

Después de todo esto, para evaluar las imágenes que obtenemos, hay que compilar el código ofrecido por la Universidad de Berkeley. Para ello, ha sido necesario trabajar en una plataforma *Intel/Linux*, debido a que *Windows* no es soportado. Una vez compilado el código, se genera una librería de funciones que tiene que ser añadida a las librerías de *Matlab®*, puesto que estas pruebas también se realizan en este entorno.

Cabe destacar que el software evalúa algoritmos de detección de bordes y, por ello, nuestro algoritmo de post-procesado se puede ver afectado ya que no sólo depende de la segmentación por agrupamiento realizada previamente sino también de estos algoritmos de detección de bordes.

5.2.3 Resultados de las pruebas

El software de evaluación nos proporciona varias clasificaciones. Una es la clasificación de los algoritmos en relación a su grado de parecido con la detección de bordes realizada por una persona y, otra clasificación que muestra la dificultad de cada imagen para ser segmentada, de la cual se mostrarán algunos ejemplos.

A continuación, pasamos a comparar los resultados de nuestros algoritmos con los que se obtuvieron en [25] para comprobar en qué casos se mejora la segmentación añadiendo el bloque de post-procesado.

En las tablas siguientes, se muestra la clasificación de nuestros algoritmos junto a la puntuación asignada que es la *medida-F* y la puntuación que se consiguió en las pruebas desarrollas en [25]. En las filas sombreadas en rojo podemos ver con qué algoritmos se ha mejorado la segmentación.

En primer lugar, vamos a comparar las imágenes es escala de grises (tabla 5.11).

Tabla 5.11. Clasificación de los algoritmos con las imágenes en escala de grises.

Escala de grises			
Clasificación	Puntuación	Puntuación en [25]	Algoritmo
0	0.79	0.79 [0]	Personas
1	0.46	0.40 [5]	Canny
2	0.45	0.43 [3]	Sobel
3	0.45	0.43 [4]	Log
4	0.45	0.43 [2]	Prewitt
5	0.34	0.44 [1]	Roberts

Observando la tabla anterior, podemos decir que, en general, para las imágenes en escala de grises se han mejorado ligeramente los algoritmos aplicando el bloque de post-procesado. Destacan los algoritmos de Canny y Roberts. El algoritmo de Canny ha pasado de ser el último de la clasificación en [25] a ser el primero si post-procesamos las imágenes. Sin embargo, con el algoritmo de Roberts ha ocurrido justo lo contrario, ha empeorado, de estar en la primera posición ha pasado a ser el último de nuestra clasificación.

A continuación, vamos a comparar las imágenes a color. En primer lugar, para las bandas a*b (tabla 5.12) y, después, las bandas RGB (tabla 5.13).

Tabla 5.12. Clasificación de los algoritmos con las imágenes a color (bandas a*b).

Color (bandas a*b)			
Clasificación	Puntuación	Puntuación en [25]	Algoritmo
0	0.79	0.79 [0]	Personas
1	0.47	0.48 [3]	Canny
2	0.47	0.48 [4]	Sobel
3	0.45	0.48 [2]	Log
4	0.44	0.48 [5]	Prewitt
5	0.35	0.49 [1]	Roberts

Para este espacio de color, no ha mejorado ningún algoritmo.

Tabla 5.13. Clasificación de los algoritmos con las imágenes a color (bandas RGB).

Color (bandas RGB)			
Clasificación	Puntuación	Puntuación en [25]	Algoritmo
0	0.79	0.79 [0]	Personas
1	0.49	0.43 [5]	Canny
2	0.47	0.45 [2]	Log
3	0.47	0.45 [1]	Sobel
4	0.45	0.44 [3]	Prewitt
5	0.33	0.43 [4]	Roberts

En este caso, para el espacio de color RGB, sucede algo parecido a lo que ocurre con las imágenes a escala de grises. En general, se mejoran todos los resultados, sobre todo los obtenidos mediante el algoritmo de detección de bordes de Canny. Mientras que el método Roberts se empeora significativamente.

La segmentación depende enormemente del espacio de color que elegimos y, por tanto, el post-procesado también. Observando las tablas anteriores, podemos decir que, para las imágenes a color, nuestros algoritmos funcionan de forma parecida respecto a los métodos de detección para ambos espacios de color mientras que, si comparamos con los resultados de [25], nuestros resultados mejoran si se trata del espacio de color RGB.

En líneas generales, los resultados que obtenemos, para todos los casos, oscilan en el mismo rango de valores. Esto es, alrededor de 0.34 para el método de detección de bordes de Roberts y, en el rango [0.44, 0.49] para el resto de algoritmos.

Con todo esto, podemos concluir que el método que realmente necesita del bloque de post-procesado es el algoritmo de Canny, que aunque con las componentes de color a*b no haya mejorado, sí ha pasado a encabezar la lista. Mientras que, para el método Roberts nuestro bloque no es necesario sino redundante debido a que empeora notablemente los resultados.

Pero, si comparamos nuestros algoritmos con los resultados de la segmentación realizada por personas (F = 0.79), todavía se alejan en gran medida.

Se comprobó si aumentando el número de clases en las que segmentábamos la imagen se mejoraba los resultados. Aumentamos a cuatro clases, en vez de tres, y los resultados oscilaban entre los valores [0.32, 0.47]. Así pues, los resultados son muy parecidos que con tres clases pero, ligeramente menores. Por lo que aumentando el número de clases no produce ningún efecto sobre los resultados.

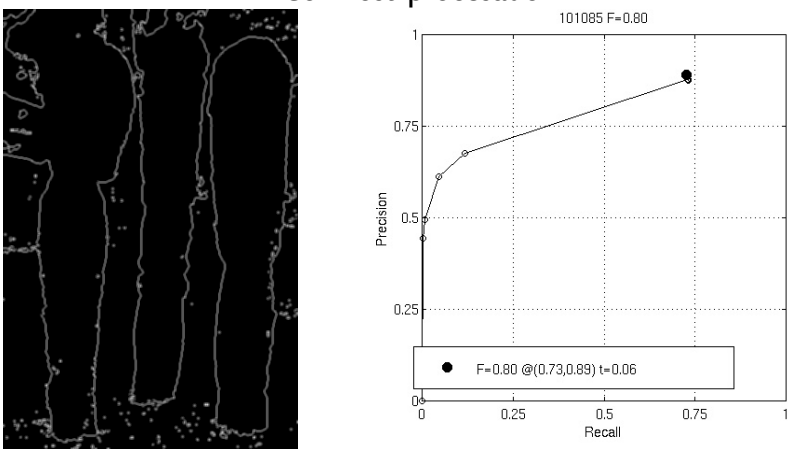
Como hemos comentado anteriormente, otra clasificación que nos ofrece la herramienta de evaluación es de las imágenes de la base de datos en función de la dificultad que tienen los algoritmos para detectar los bordes. En algunas imágenes, la *medida-F* que se obtiene supera en gran proporción a la media obtenida por los algoritmos.

A continuación, se van a analizar y comparar con los resultados obtenidos en [25] algunos ejemplos. Hay que señalar que en las pruebas realizadas en [25] se realizó la segmentación en dos regiones mientras que, nosotros hemos aumentado el número de clases a tres y, después, hemos aplicado el bloque de post-procesado.

Uno de los mejores resultados que se obtienen, al igual que en [25], es con la imagen mostrada en la figura 5.39. Los objetos de la imagen tienen un color uniforme y diferenciado del fondo. Los resultados se mejoran en 0.05 respecto a [25], sólo que con otro método de detección. En nuestro caso, el mejor resultado con F= 0.80 se obtiene mediante el método de Sobel con las componentes a*b y, no con el algoritmo de detección de Roberts como en [25] con F= 0.75 (ver figura 5.40). Por tanto, esto es un ejemplo excepcional ya que los mejores resultados se obtienen con las componentes a*b y no con el espacio RGB, como sucede en media con el resto de imágenes.

Figura 5.39. (Izqda.) Imagen #17 de la base de datos y (dcha.) bordes obtenidos por personas F=0.93.

Con Post-procesado



Sin Post-procesado

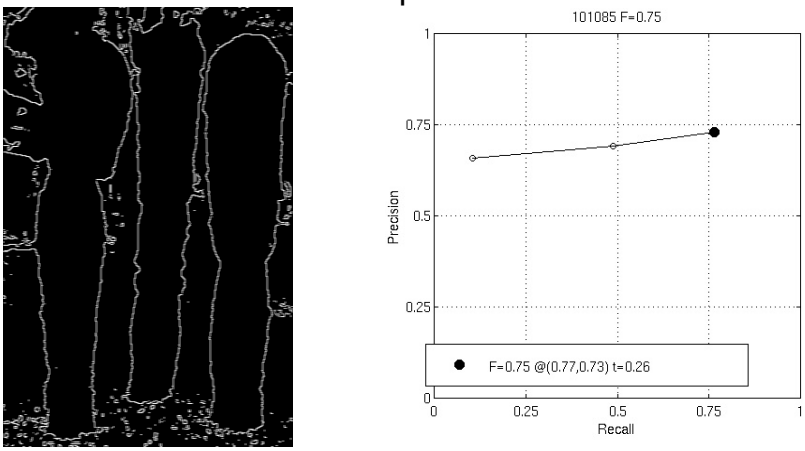


Figura 5.40. (Arriba) Resultado método Sobel con el post-procesamiento F=0.80 y (abajo) resultado método Roberts sin post-procesamiento F=0.75 (*Ref.[25]*), para la imagen #17.

Otro caso interesante, se consigue con la imagen #97 de la base de datos (figura 5.41). Este ejemplo sigue la media y el mejor resultado se consigue con las componentes RGB y el método de detección de bordes Log. Este ejemplo es interesante ya que se puede apreciar la reducción de la sobre-segmentación al aplicar el post-procesado (figura 5.42).

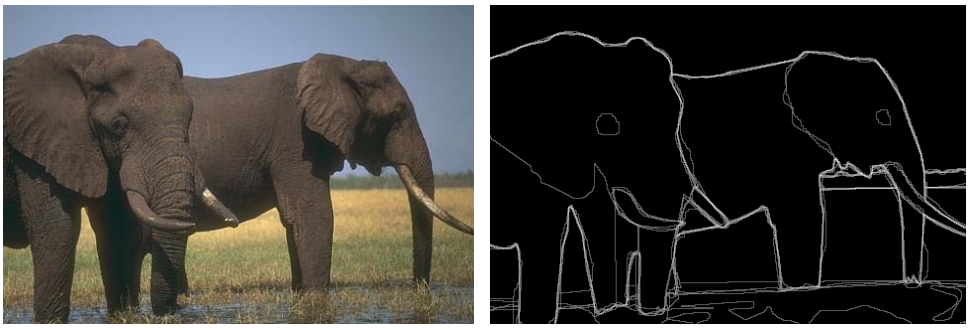
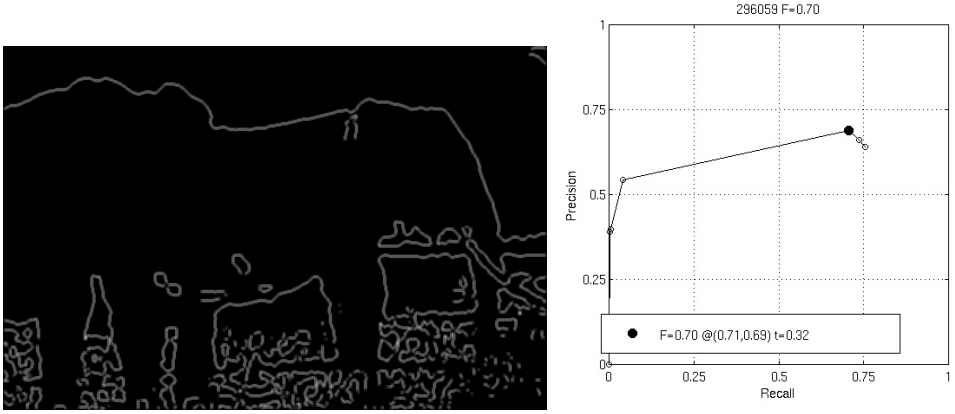


Figura 5.41. (Izqda.) Imagen #97de la base de datos y (dcha.) bordes obtenidos por personas F=0.88.

Con Post-procesado



Sin Post-procesado

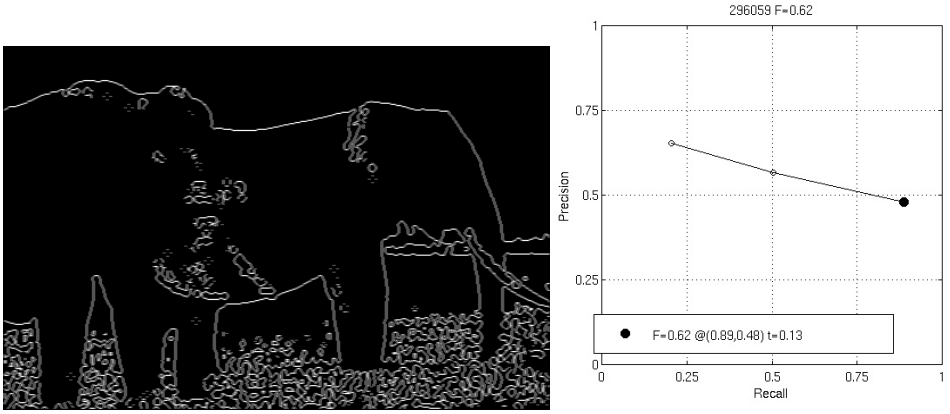


Figura 5.42. (Arriba) Resultado método Log con el post-procesamiento F=0.70 y (abajo) resultado método Log sin post-procesamiento F=0.62 (*Ref. [25]*), para la imagen #97.

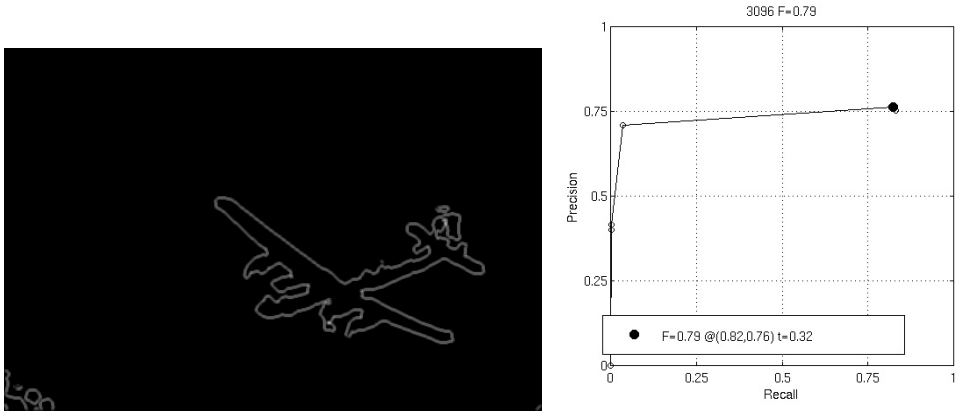
En este caso, el resultado se ha mejorado 0.08 debido a que algunas regiones se han fusionado ocasionando regiones más compactas. Podemos ver como en la región perteneciente al suelo se ha reducido la sobre-segmentación.

Otro ejemplo de mejora de la segmentación se consigue con la imagen #65 (figura 5.43) de la base de datos en escala de grises. En nuestros resultados, el algoritmo de detección de bordes Log consigue la mejor puntuación con F= 0.79 mientras que en [25], encabeza la lista también el método Log pero con una puntuación de F= 0.44. Esto supone una diferencia de 0.35. Se pueden observar los resultados en la figura 5.44, en la que vemos que se ha unificado totalmente el fondo, el cielo, y que, se distingue totalmente el objeto, el avión.

Esta mejora se debe a que nuestros algoritmos fueron diseñados para imágenes con un objeto definido y, como vemos en la figura 5.43, este es el caso de una imagen con el objeto diferenciado del fondo.

Figura 5.43. (Izqda.) Imagen #65 de la base de datos y (dcha.) bordes obtenidos por personas F=0.94.

Con Post-procesado



Sin Post-procesado

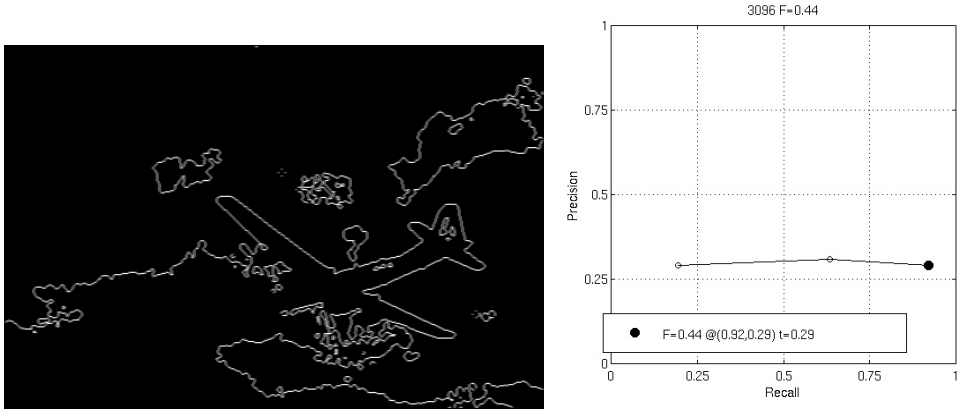


Figura 5.44. (Arriba) Resultado método Log con el post-procesamiento F=0.79 y (abajo) resultado método Log sin post-procesamiento F=0.44 (*Ref. [25]*), para la imagen #65.

5.2.4 Conclusiones de las pruebas automáticas

A lo largo del apartado hemos visto cómo, en la mayoría de los casos, se mejora la segmentación si aumentamos el número de clases y aplicamos el post-procesado. Como hemos mencionado, el post-procesado se ve afectado por varios factores como por el espacio de color, el tipo de segmentación, la detección de bordes, etc.

Por todo ello, nuestros resultados se pueden ver empeorados. Aunque si los comparamos con los que obtienen otros algoritmos que han sido evaluados con la herramienta de la Universidad de Berkeley, publicados en [26], comprobamos que la calificación que hemos obtenido está muy por debajo. Estos algoritmos consiguen hasta un 0.70 mientras que, los que hemos evaluado en este proyecto

no superan el 0.50. En [26] se puede localizar la descripción de algunos de estos algoritmos. Podemos observar los resultados de estos algoritmos en la figura 5.45.

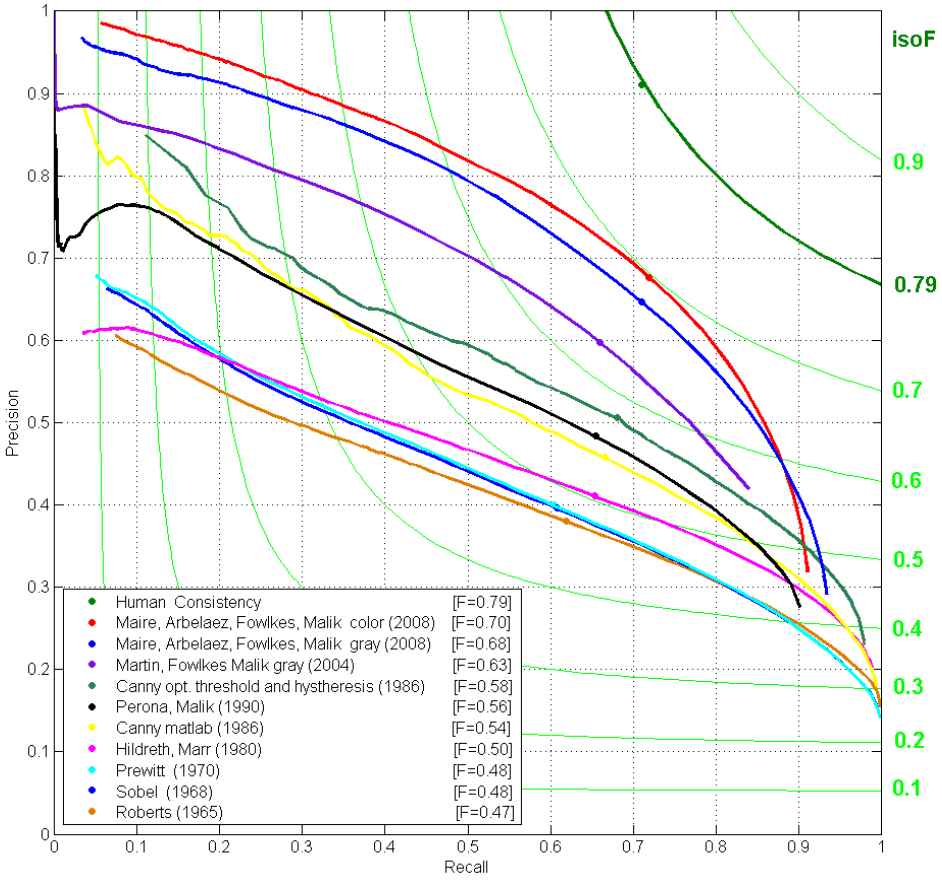


Figura 5.45. Evaluación de los algoritmos publicados (*Ref. [27]*).

Todavía lejos de encontrar un algoritmo que consiga unos resultados tan buenos como los que se muestran en la figura anterior, en la mayoría de los casos evaluados se ha mejorado los algoritmos de segmentación realizados en [25] añadiendo nuestro bloque de post-procesamiento.

CAPÍTULO 6

CONCLUSIONES Y LÍNEAS FUTUTRAS

Este capítulo es el cierre de este trabajo en el que presentamos las conclusiones que hemos obtenido de este proyecto y las líneas de desarrollo futuras para la mejora y continuación de este sistema.

6.1 CONCLUSIONES

El propósito antes de la realización de este proyecto era la implementación de un bloque de post-procesamiento de regiones en imágenes segmentadas que fuera capaz de reducir la sobre-segmentación de algunos algoritmos de segmentación.

En general, las fases que se siguieron para la investigación y desarrollo de este sistema se corresponden con los bloques de la aplicación, tal y como se puede ver en el siguiente esquema general.

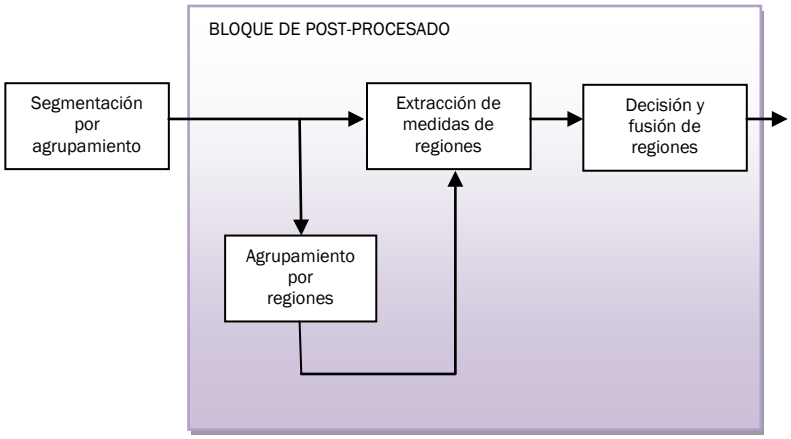


Figura 6.1. Esquema general de un bloque de post-procesamiento.

El primer paso de este trabajo fue la elección de los parámetros necesarios para la descripción de regiones. Éstos tenían que poder extraer información de regiones de forma independiente a las características que presenta un objeto. De forma que se eligieron varios descriptores que en su conjunto pudieran extraer información de imágenes de naturaleza diversa, lo que supone una dificultad alta. Así pues, y si el usuario pretende estudiar imágenes con características similares, los algoritmos permiten el uso de uno o más parámetros de forma independiente. También se procedió a la normalización de estos parámetros para que ninguno tuviera más peso que otro en la decisión.

Una vez obtenidos los descriptores de las regiones, se diseño un decisor capaz de diferenciar cuándo era necesaria realizar la fusión de una o más regiones. Para ello, hubo que encontrar el umbral necesario para esta decisión. Después de

varias pruebas, se decidió utilizar un umbral propio para cada imagen de forma que la decisión dependiera de la imagen que se estaba post-procesando y así poder estudiar cualquier tipo de imagen.

Después de realizar todos los algoritmos de la herramienta, se implementó una interfaz gráfica capaz de ofrecer flexibilidad al usuario con total libertad en la elección de parámetros. Además, esta interfaz también da facilidad para escoger el método que se quiera utilizar para la segmentación, integrando una librería de algoritmos de segmentación realizada en otros trabajos o pudiendo cargar una imagen ya segmentada.

Nuestros algoritmos funcionan con casi cualquier método de segmentación. El problema que hemos encontrado es que existe un método con el que nuestra aplicación no funciona, este es el algoritmo de segmentación watershed por gradiente. El algoritmo de watershed por gradiente tiene como característica principal ofrecer un grado muy alto de sobre-segmentación y debido al tamaño máximo para un array soportado por *Matlab®* no es admisible que la imagen segmentada contenga un número de regiones muy elevado ya que se hace imposible calcular el número de combinaciones posibles.

A pesar de esto, a lo largo del capítulo 5 de esta memoria se ha demostrado que en la mayoría de los casos, nuestra herramienta mejora la segmentación unificando regiones sobre-segmentadas de algoritmos más sencillos, como es el caso del algoritmo de segmentación por agrupamiento *kmeans*. Aunque hay que destacar que la segmentación depende enormemente de la extracción de características previas de la imagen como el color o la textura y, por tanto, el resultado final de nuestra aplicación también depende de lo buena o mala que sea la segmentación.

Así pues, podemos decir que las necesidades expuestas al principio de este proyecto han sido cumplidas.

Pero, la intención de este trabajo no es finalizar una línea de trabajo, sino el de empezar una vía de desarrollo hacia un sistema de post-procesado más sofisticado capaz de funcionar con todos los algoritmos de segmentación

existentes. En este sentido, se plantea el siguiente apartado en el que se exponen algunas ideas de líneas futuras de investigación para que, de este modo, este sistema sirva de plataforma para futuros desarrollos.

6.2 LÍNEAS FUTURAS

En la realización de este proyecto hemos identificado una serie de aspectos susceptibles de modificación total o parcial en futuros desarrollos del bloque de post-procesado.

Una evolución parcial de nuestro sistema sería la modificación del algoritmo de combinación que se desarrolló. Esta reforma consistiría en eliminar de la estructura obtenida el par de regiones que no son adyacentes ya que nunca se utilizan pero, sí se calculan los parámetros de estas combinaciones ocasionando un aumento del tiempo de cómputo y una pérdida de eficiencia en el sistema. Para ello, se podría utilizar un esquema similar al de una estructura RAG, explicada en el capítulo 2 de esta memoria.

Además, continuando con esta línea, aumentaría el número de clases soportadas por el algoritmo de combinaciones, pudiendo ser utilizada la herramienta con el algoritmo watershed por gradientes.

Otra mejora, que podría ser interesante, es que los parámetros fueran acompañados de unos pesos en el vector de características y que el usuario pudiera modificarlos a su conveniencia.

Otra línea de desarrollo sería aumentar el grado de flexibilidad de la herramienta. Esto es, que primero se pudiera hacer un post-procesado atendiendo a un/os parámetro/s y, después a otros, refinando así el post-procesado que sufriría la imagen segmentada.

Como se comentó en capítulos anteriores, el post-procesado tiene una unión muy fuerte con la segmentación. La herramienta que hemos desarrollado puede ser utilizada para casi cualquier algoritmo de segmentación con lo que se podría crear unos algoritmos de post-procesado atendiendo de forma más específica a cada una de las necesidades de cada algoritmo de segmentación.

Un cambio en este trabajo, no de la herramienta software implementada, sería realizar las pruebas automáticas de forma más exhaustiva, dando intensidad a cada borde que se une en cada nivel de fusión. Es decir, la frontera de cada

región que se une se mantendría pero con una ponderación. Cada frontera iría adquiriendo más peso en función del nivel de fusión, siendo la frontera con más peso la frontera final calculada.

ANEXO A

ESTRUCTURA DEL CÓDIGO

A.1 INTERFAZ GRÁFICA

A.1.1 Funciones

La función **Postprocess.m** es la función principal y se corresponde a la interfaz gráfica. Se compone de las siguientes funciones:

- **Postprocess_OpeningFcn(hObject, eventdata, handles, varargin).** Función encargada de crear todas las variables globales del sistema.

- **varargout = Postprocess_OutputFcn(hObject, eventdata, handles).** Función de salida de datos del sistema.

- **BotonCargarImagen_Callback(hObject, eventdata, handles).** Función encargada de controlar los eventos cuando se carga una imagen nueva y gestionar los elementos gráficos para cargarla.

- **EtiqRegiones_Callback(hObject, eventdata, handles).** Función que realiza el paso de una cadena de caracteres a un número para el posterior uso en el algoritmo *kmeans*.

- **BotonSegmentar_Callback(hObject, eventdata, handles)**. Función encargada de realizar la segmentación y de gestionar los elementos gráficos necesarios. Utiliza las siguiente función auxiliar:

    - **Xsegmentada = Segmentakmeans(imageOrig, regiones)**. Función que realiza la segmentación de la imagen cargada, *imageOrig*, mediante el algoritmo *kmeans* con el número de regiones introducido por el usuario.
    - **Xsegmentada = main_gui(dirección_Imagen_original)**[12]. Función que realiza otros tipos de segmentación a través de otra interfaz gráfica.

[12] Estructura de la interfaz main_gui en [25].

- **CB_xxxx_Callback(hObject, eventdata, handles)**. Funciones encargadas de gestionar que tipo de segmentación desea realizar el usuario.

- **RBxxxxx_Callback(hObject, eventdata, handles)**. Funciones encargadas de gestionar si un parámetro es seleccionado o no.

- **BotonPostProcesar_Callback(hObject, eventdata, handles)**. Función que realiza el post-procesado de la imagen. Esto es, calcula los parámetros para cada región individual o combinada y realiza la decisión con su posterior fusión. Usa las siguientes funciones auxiliares:

    - **nr = numRegiones(image)**. Función que calcula el número de regiones de la imagen introducida como parámetro (*image*).
    - **Regiones = regionesIndividuales(image)**. Esta función se encarga de binarizar cada región simple de la imagen segmentada introducida (*image*). Devuelve una estructura con tamaño [2xnúmero de regiones] que guarda la imagen binarizada y la región a la que corresponde.
    - **Regiones = agrupamiento(image)**. Función que realiza las combinaciones posibles para fusionar. Devuelve una estructura con tamaño $[2x(2^{nr} - nr - 2)]$ siendo nr el número de regiones, guarda la imagen binarizada resultante de cada combinación y las regiones que se fusionan.
    - **num = compConectadas(image)**. Función encargada de calcular el número de componentes conectadas de cada región.
    - **p = perimeter(image)**. Esta función realiza el cálculo del perímetro de cada región.
    - **NI = newImage(image, regionesFusionadas)**. Esta función calcula la nueva imagen (*NI*) resultante de la fusión de dos o más regiones.
    - **pintaBar**. Esta función se encarga de realizar la gráfica del histograma cuando la decisión final se ha realizado.

Funciones del menú:

- **CargarImagen_Callback(hObject, eventdata, handles)**. Función encargada de cargar una imagen y gestionar los elementos gráficos para cargarla.

- **Salir_Callback(hObject, eventdata, handles)** . Función encargada de cerrar la aplicación.

- **ImagenSegmentada_Callback(hObject, eventdata, handles)** . Función que guarda la imagen segmentada en un archivo con extensión *.jpeg*.

- **ImagenPostProcesada_Callback(hObject, eventdata, handles)** . Función que guarda la imagen post-procesada en un archivo con extensión *.jpeg*

- **borrarIS_Callback(hObject, eventdata, handles)** . Función encargada de borrar la imagen segmentada y gestionar todos los elementos gráficos que el borrado conlleva.

- **borrarIP_Callback(hObject, eventdata, handles)** . Función encargada de borrar la imagen post-procesada y gestionar todos los elementos gráficos que el borrado conlleva.

- **borrarT_Callback(hObject, eventdata, handles)**. Función encargada de borrar todo y gestionar todos los elementos gráficos que el borrado conlleva.

A.1.2 Diagramas de bloques

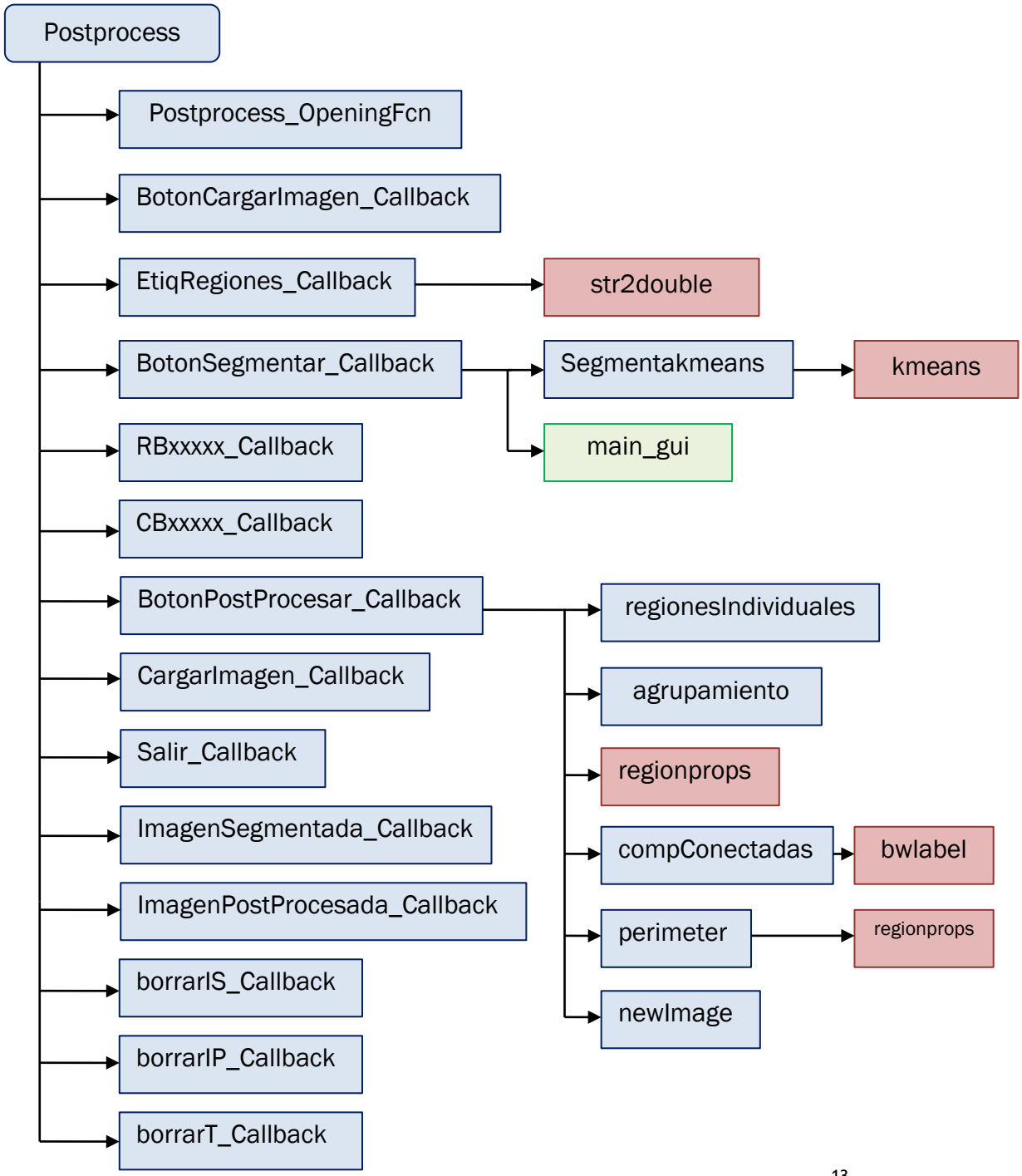


Figura A.1. Diagrama de bloques de la librería de funciones principal.[13]

13 Las funciones de implementación propia se muestran en color azul y las procedentes de la librería de procesado de imágenes de Matlab® se muestran en rojo. La función coloreada en verde pertenece a la librería de segmentación desarrollada en [25].

A.2 PRUEBAS AUTOMÁTICAS

A.2.1 Funciones

Para el desarrollo de las pruebas automáticas, se han creado una serie de funciones basadas en las funciones que se crearon en [25] para la ejecución de las mismas pruebas. En este caso, no se ha desarrollado una interfaz gráfica de usuario debido a que estas funciones son muy sencillas.

Estas funciones se componen, principalmente, de funciones de *Matlab®* dirigidas a la lectura de la base de datos y la escritura de las imágenes resultantes.

- **dataload(type, algorithm, format).** Función encargada de cargar las imágenes de test o train (*type*), realizar el algoritmo de agrupamiento *kmeans,* el post-procesamiento y finalmente, aplicar el algoritmo de detección de bordes (*algorithm*) sobre las características de color determinadas (*format*). Esta función utiliza las siguientes funciones auxiliares.
    - **bsdsRoot.m.** Función que indica la ruta donde se encuentra la carpeta con las imágenes de la base de datos.
    - **[idds] = imgList(type).** Función que devuelve una lista con los identificadores de las imágenes (*idds*) del conjunto de entrenamiento o test (*type*).
    - **[filename] = imgFilename(iid).** Devuelve el nombre completo de la imagen (*filename*) a partir de un identificador.
    - **[im] = imgRead(iid, format).** Función que devuelve la imagen de un identificador determinado (*iid*) en color o en escala de grises (*format*).
    - **I = kmedias(image).** Realiza el algoritmo de segmentación *kmeans* a la imagen introducida (*image*) , devuelve una nueva imagen (*I*) segmentada.
    - **imagenPostprocesada = decisor(varargin).** Función que devuelve la imagen post-procesada (*imagenPostprocesada*) y que se le pasa por argumentos (*varargin*), primero, la imagen segmentada y, a continuación, los parámetros que se desean utilizar.

A.2.2 Diagrama de bloques

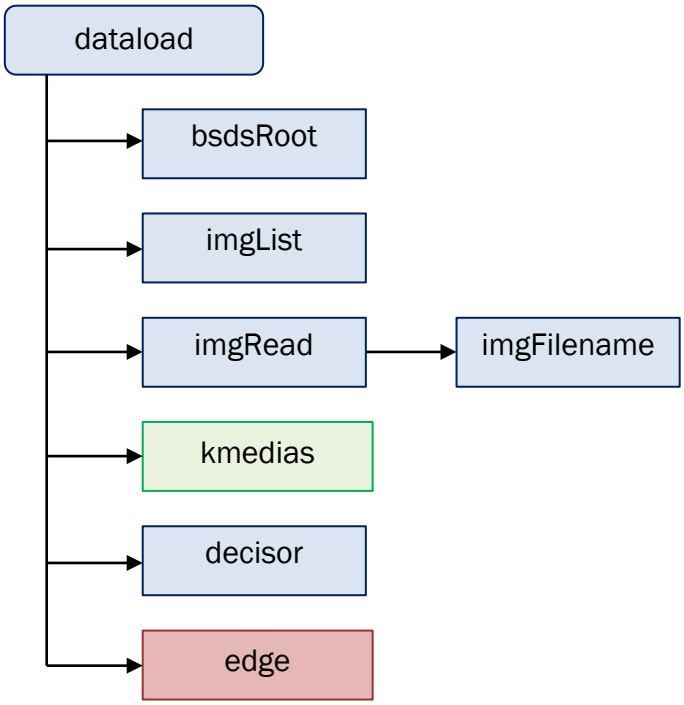


Figura A.2. Diagrama de bloques de la librería de funciones para las pruebas automáticas.

ANEXO B

MANUAL DE USUARIO

B.1 MANUAL DE USUARIO DE LA APLICACIÓN

La ejecución de la aplicación se inicia introduciendo en la línea de comandos de *Matlab®* el nombre del programa principal, *Postprocess*. Así, se abrirá la ventana principal (figura B.1).

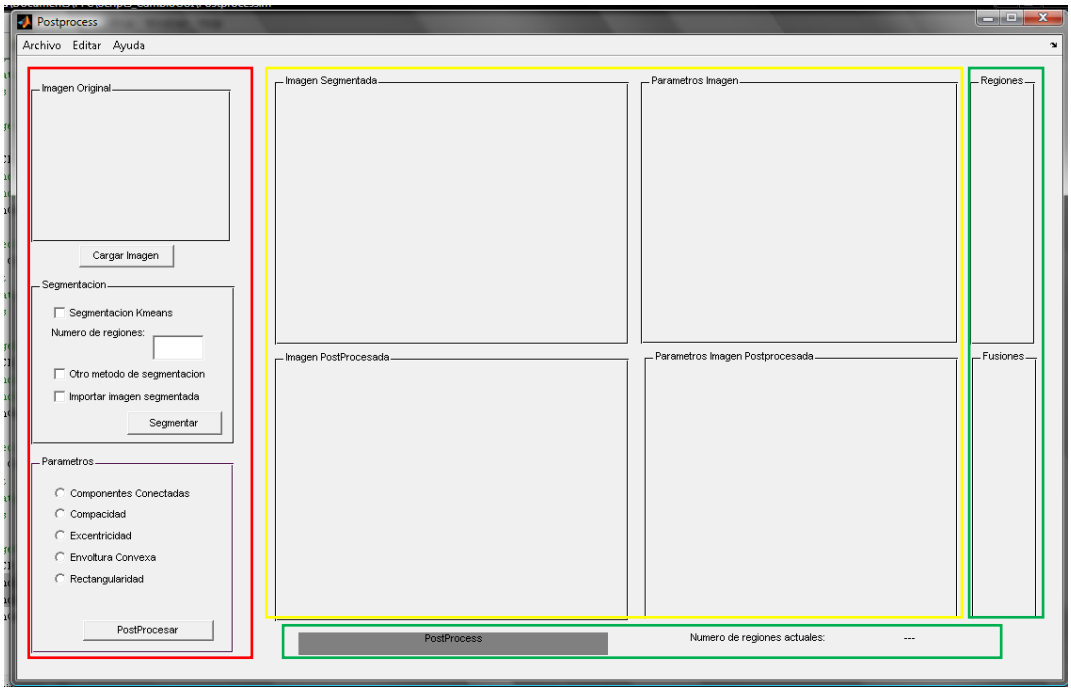


Figura B.1. Ventana principal de la aplicación.

La ventana principal se puede dividir en tres partes. Según la figura B.1, el recuadro rojo se corresponde con la parte donde se va realizando la ejecución paso a paso (cargar imagen, segmentar y post-procesar), el amarillo con la visualización de resultados y el verde para información adicional.

Pasamos a explicar cada una de las partes por separado.

Cargar Imagen:

Para cargar una imagen encontramos dos formas diferentes de realizarlo. La primera es mediante el botón situado en la parte superior de la izquierda de la interfaz:

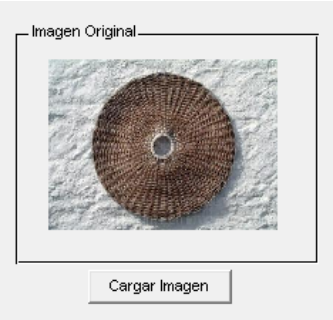


Figura B.2. Módulo de la interfaz para cargar una imagen.

El otro modo que tenemos para cargar una imagen es mediante la opción *Cargar Imagen* del menú *Archivo*. La forma abreviada de realizar la carga de una imagen se ejecuta mediante *Ctrl+C*.

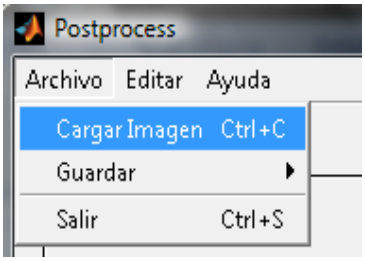


Figura B.3. *Archivo->Cargar Imagen*.

Una vez realizada la selección, aparecerá una pantalla de selección de archivo (figura B.4).

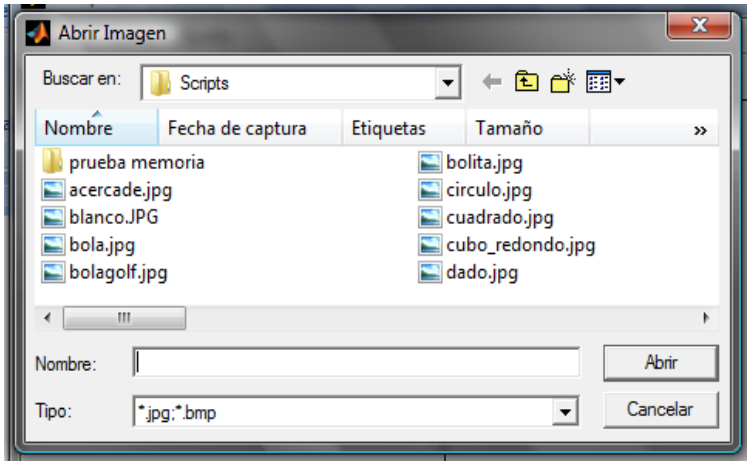


Figura B.4. Ventana de selección de archivo.

Se selecciona la imagen que se desea y se pasa al siguiente módulo.

Segmentar Imagen o Cargar Imagen Segmentada:

En el panel de segmentación (figura B.5) nos encontramos con tres opciones. Una es realizar la segmentación a través del algoritmo *kmeans*, otra realizar otro método de segmentación y otra cargar una imagen segmentada.

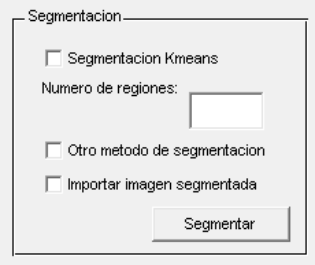


Figura B.5. Panel de Segmentación.

Si decidimos realizar la segmentación a través del método kmeans, debemos seleccionar la correspondiente opción e introducirle el número de regiones con el que queremos segmentar la imagen cargada.

Si queremos realizar otro método de segmentación, tendremos que seleccionar la opción "*Otro método de segmentación*" y nos aparecerá otra ventana para realizar la segmentación (figura B.6) con la imagen que ya habíamos cargado. Seleccionaremos el método que deseemos[14] y le daremos al botón *Salir*.

[14] Manual de usuario de la herramienta de segmentación en [25].

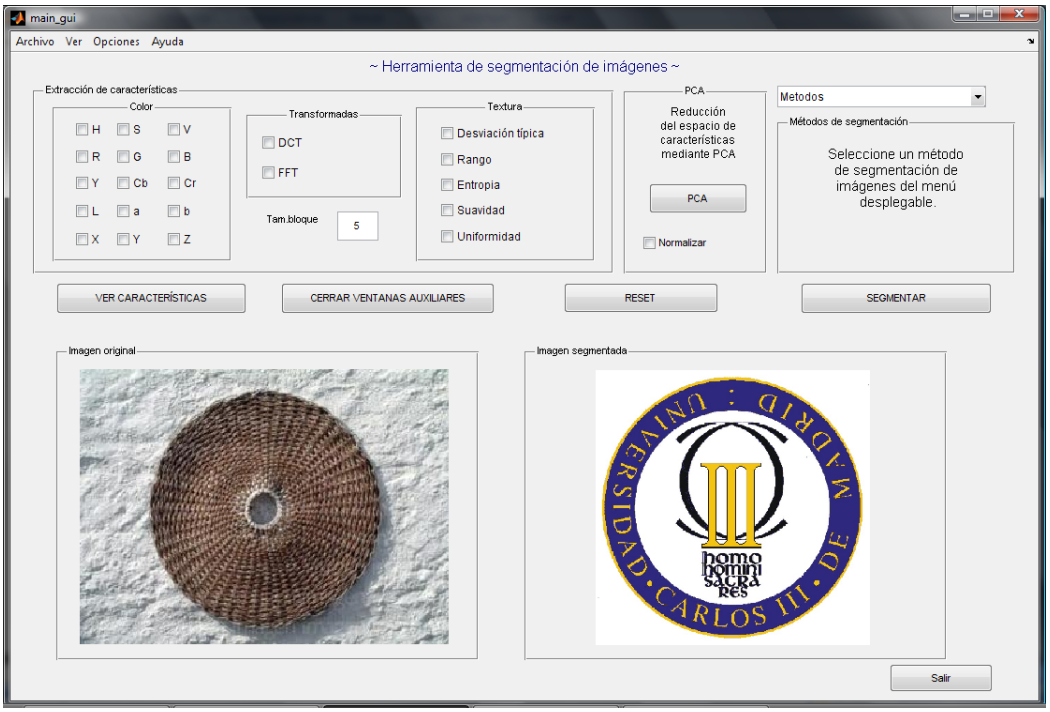


Figura B.6. Ventana de la herramienta de segmentación.

Por el contrario, si disponemos de una imagen segmentada podemos saltar el paso anterior de cargar una imagen original y cargar directamente esta imagen segmentada. Es importante señalar que la imagen segmentada se ha tenido que guardar en un archivo *.mat*, ya que si no *Matlab*® la convierte en una imagen RGB y, por lo tanto, es imposible etiquetarla.

La imagen segmentada se visualizará en el panel correspondiente (figura B.7).

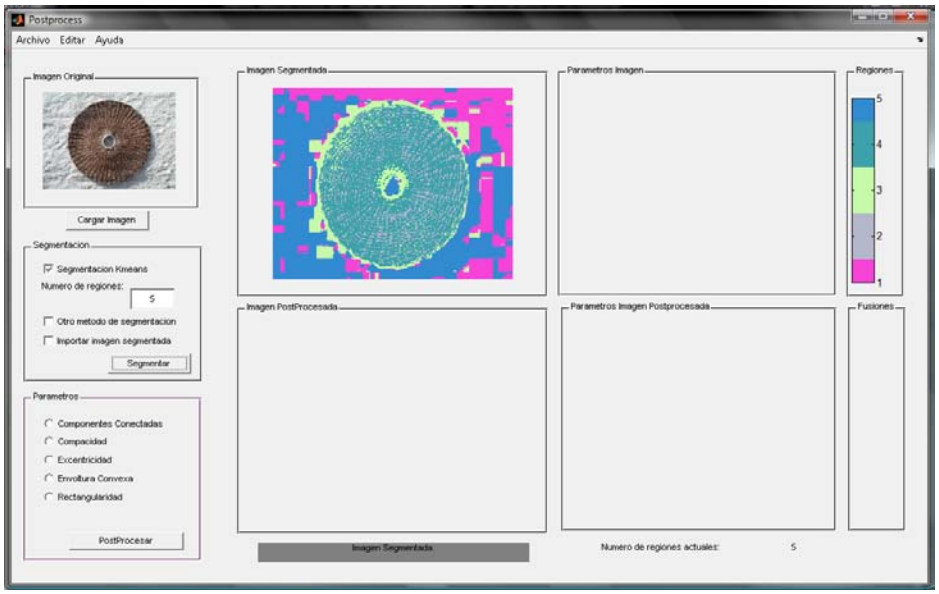


Figura B.7. Ventana de la aplicación tras haber segmentado la imagen.

Como podemos observar en la anterior figura, a la derecha de la ventana aparece una barra que nos indica con qué región se corresponde cada color. Esto nos servirá de ayuda para cuando se realice el post-procesado ir viendo paso a paso que regiones se están fusionando.

También vemos que abajo a la derecha de la ventana de la aplicación hay un campo que nos informa del número de regiones actuales. Este campo igualmente sirve de ayuda para el post-procesamiento.

Elección de parámetros y post-procesado:

En el panel de Parametros (figura B.8), el usuario puede hacer la elección de los parámetros que quiere utilizar para el post-procesado de la imagen segmentada.

Figura B.8. Panel de Parámetros.

Una vez decidido los parámetros, se debe pulsar el botón de Postprocesar para que comience el post-procesado. En la siguiente figura vemos el resultado de pulsar el botón.

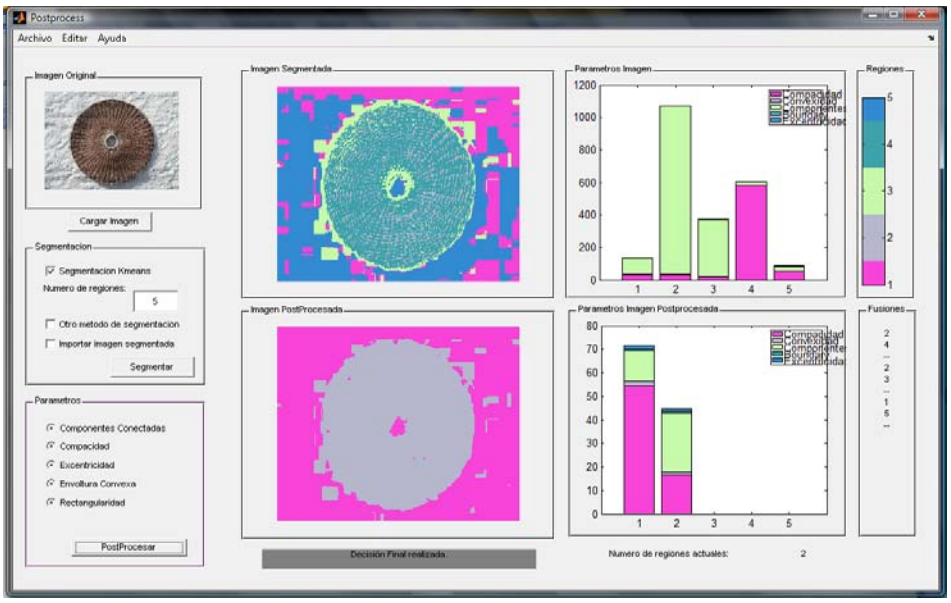


Figura B.9. Ventana de la aplicación tras haber post-procesado la imagen.

En las gráficas de la derecha se pueden observar, arriba los parámetros iniciales de la imagen original y abajo, los parámetros de imagen post-procesada.

Vemos como en el panel Fusiones (figura B.10) aparecen las fusiones que se han ido realizando a lo largo de la ejecución del algoritmo. Cada paso que ha realizado el algoritmo está separado por unas rayitas (- -).

Figura B.10. Panel de fusiones.

En la zona sombreada de gris se puede conocer el estado en el que se encuentra la ejecución de la aplicación.

Menú:

Guardar:

En el menú *Archivo* se pueden guardar los resultados obtenidos en la segmentación y en el post-procesado. Ver figura B.11.

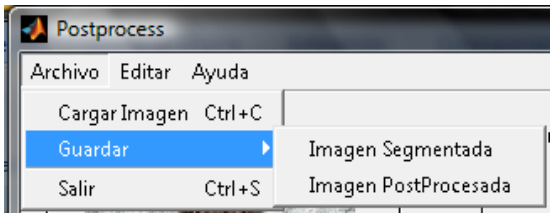


Figura B.11. *Archivo->Guardar*.

Salir:

Para salir de la aplicación encontramos una opción en el menú *Archivo*. La forma abreviada es *Ctrl*+S. Figura B.12.

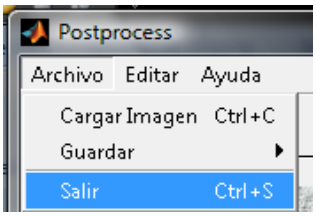


Figura B.12. *Archivo->Salir*.

Editar:

En el menú *Editar* podemos borrar algún paso de la aplicación, bien la imagen segmentada, la imagen post-procesada o borrar todo para resetear la aplicación.

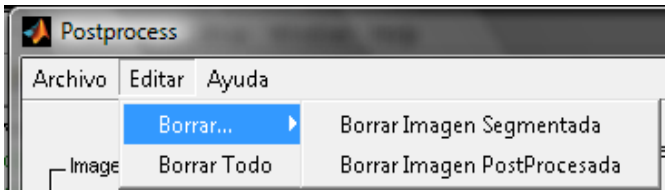


Figura B.13. *Editar*.

Ayuda:

En *Ayuda* podemos ver detalles de la versión en la opción *Acerca de...*. Figuras B.14 y B.15.

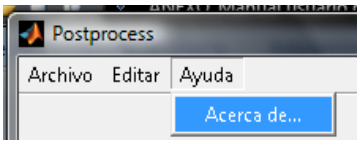


Figura B.14. *Ayuda*.

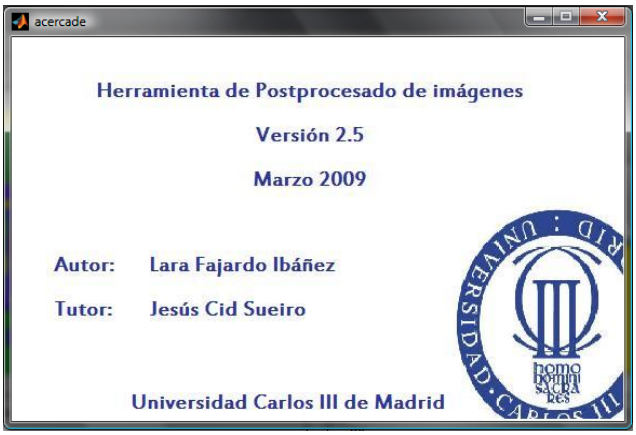


Figura B.15. *Ayuda->Acerca de...*.

BIBLIOGRAFÍA

[1] Pajares, G., de la Cruz, J.M., *Visión por computador, imágenes digitales y aplicaciones* (2001). Editorial RA-MA.

[2] González, R.C., Wood, R.E., *Digital Image Processing, Third Edition* (2008). Addison-Wesley.

[3] González Jimenez, J., *Visión por computador* (2000). Madrid Paraninfo.

[4] González, R.C., Wood, R.E., *Digital Image Processing using Matlab* (2004). Addison-Wesley.

[5] Smith, Scott T. *Matlab: Advanced GUI Development* (2006). Dog Ear Publishing.

[6] Pajares, G., de la Cruz, J.M., *Ejercicios resueltos de visión por computador* (2007). Editorial RA-MA.

[7] *The MathWorks.* www.mathworks.com.

[8] Uriel H. Hernández-Belmonte, V. Ayala-Ramírez, and Raúl E. Sánchez-Yánez, *Una alternativa para el etiquetado de componentes conectadas usando vecindades*. Universidad de Guanajuato, Campus Salamanca FIMEE. Laboratorio de Visión Robótica e Inteligencia Artificial.

[9] Muñoz Perez, J., *Representación de formas y descripción* (2006). Universidad de Málaga.

[10] Cid Sueiro, J., *Transparencias asignatura Tratamiento Digital de Imagen* (2007). Ingeniería de Telecomunicación: Sonido e Imagen. Universidad Carlos III de Madrid

[11] Curso teórico de procesado de imágenes digitales y Topología digital, Universidad de Sevilla. http://www.sav.us.es/formaciononline/asignaturas/asigpid/

[12] Jia-Nan Wang, Jun Kong, Ying-Hua Lu, Wen-Xiang Gu, Ming-Hao Yin, Yong-Peng Xiao, *A region-based SRG algorithm for color image segmentation* (2007). College of computer Science.

[13] Silvia Alayón, José Luis Sánchez, José F. Sigut, Juan A. Méndez, *Segmentación automática de núcleos solapados en imágenes de citologías*. Departamento de Ingeniería de Sistemas y Automática, y Arquitectura y Tecnología de Computadores, Universidad de La Laguna, Tenerife, España

[14] Yuan He, Yupin Luo, Dongcheng Hu, *Seeded region Merging based on gradient vector flow for image segmentation* (2006). Departament of Automation, Tsinghua University, Beijing.

[15] Frank Y. Shih, Shouxian Cheng, *Automatic seeded region growing for color image segmentation* (2005). Computer vision laboratory, College of Computing Sciences, New Jersey Institute of Technology.

[16] Haris K., Efstratiadis S.N. y Maglaveras N, *Watershed-based image segmentation with fast region merging* (1998). Lab of Medical Informatics, Faculty of Medicine Aristotle University.

[17] Haris K., Efstratiadis S.N., Maglaveras N. y Katsaggelos A.K., *Hybrid Image Segmentation Using Watersheds and Fast Region Merging* (1998).

[18] Hernandez S. E. y Barner K. E., *Joint region merging criteria for watershed-based image* segmentation. Department of Electrical and Computer Engineering, University of Delaware, Newark.

[19] Javier Jesús Gutiérrez Rodríguez, *Medida de la rectangularidad*. Universidad de Sevilla.

[20] Diego Orlando Barragán Guerrero, *Manual de interfaz gráfica de usuario en Matlab*. Universidad técnica particular de Loja.

[21] The Mathworks, *MATLAB 7 Creating Graphical User Interfaces*.

[22] M. C. José Jaime Esqueda Elizondo, *Interfaces Gráficas en Matlab Usando GUIDE*. Universidad Autónoma de Baja California, Unidad Tijuana.

[23] Wei, S., Zhao Y., Zhu Z., *Meaningful Regions Segmentation in CBIR* (2005). Institute of Information Science, Beijing Jiaotong University.

[24] Luo, J., Guo, G., *Perceptual grouping of segmented regions in color images* (2003). University of California.

[25] Jiménez Vazquez, E. *Desarrollo de una librería de algoritmos de segmentación de imágenes* (2008). Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid.

[26] *The Berkeley Segmentation Dataset and Benchmark*, Base de datos de imágenes segmentadas. Universidad de Berkeley. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[27] Marie, M., Arbeláez, P., Fowlkes, C., Malik, J. *Using Contours to Detect and Localize Junctions in Natural Images*. Universidad de California, Berkeley. Universidad de California, Irvine.